Injecting Inductive Biases into Distributed Representations of Text

Victor Prokhorov

Department of Theoretical and Applied Linguistics

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

Selwyn College August 2021

Declaration

I hereby declare that my dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University of similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Victor Prokhorov

August 2021

Injecting Inductive Biases into Distributed Representations of Text

Victor Prokhorov

Abstract

Distributed real-valued vector representations of text (a.k.a. embeddings), learned by neural networks, encode various (linguistic) knowledge. To encode this knowledge into the embeddings the common approach is to train a large neural network on large corpora. There is, however, a growing concern regarding the sustainability and rationality of pursuing this approach further. We depart from the mainstream trend and instead, to incorporate the desired properties into embeddings, use *inductive biases*.

First, we use Knowledge Graphs (KGs) as a *data-based* inductive bias to derive the semantic representation of words and sentences. The explicit semantics that is encoded in a structure of a KG allows us to acquire the semantic representations without the need of employing a large amount of text. We use graph embedding techniques to learn the semantic representation of words and the sequence-to-sequence model to learn the semantic representation of sentences. We demonstrate the efficacy of the inductive bias for learning embeddings for rare words and the ability of sentence embeddings to encode topological dependencies that exist between entities of a KG.

Then, we explore the amount of information and sparsity as two key (*data-agnostic*) inductive biases to regulate the utilisation of the representation space. We impose these properties with Variational Autoencoders (VAEs). First, we regulate the amount of information encoded in a sentence embedding via constraint optimisation of a VAE objective function. We show that increasing amount of information allows to better discriminate sentences. Afterwards, to impose distributed sparsity we design a state-of-the-art Hierarchical Sparse VAE with a flexible posterior which captures the statistical characteristics of text effectively. While sparsity, in general, has desired computational and statistical representational properties, it is known to compensate task performance. We illustrate that with distributed sparsity, task performance could be maintained or even improved.

The findings of the thesis advocate further development of inductive biases that could mitigate the dependence of representation learning quality on large data and model sizes.

Acknowledgements

I would like to thank Professor Nigel Collier for giving me an opportunity to pursue a PhD in his group and allowing me to explore research directions that are of my interest.

My greatest gratitude goes to Mohammad Taher Pilehvar, Dimitri Kartsaklis, Ehsan Shareghi, and Yingzhen Li who at different stages of my PhD helped me shape my research and also grow as a researcher.

Also, I would like to dedicate a few extra lines to express my thankfulness to Ehsan Shareghi, who especially played a significant role in my development as a researcher and introduced me to Variational Autoencoders and Bayesian machine learning in general. Besides the research, I learned a lot from him about commitment and dedication to one's work. He pushed me to the new reaches that I would not be able to achieve on my own.

I also would like to thank my fellow students at the University of Cambridge: Costanza Conforti, Yi Zhu, Marinela Parovic, Qianchu (Flora) Liu, Fangyu Liu, Edoardo Ponti, Olga Majewska, Gamal Crichton, Oli Liu, Aliaksei Mikhailiuk, Ryan-Rhys Griffiths, Ilia Shumailov who made my study memorable and enjoyable.

I was extremely lucky to do an internship at Mila during my PhD. For this, I am grateful to Siva Reddy and his Lab. In particular would like to thank: Lucas Torroba Hennigen, Vaibhav Adlakha, Devang Kulshreshtha, Andreas Madsen, Nicholas Meade, Nathan Schucher, Amirhossein Kazemnejad, and Zichao Li. Also, a special thank to Alessandro Sordoni who agreed to co-supervise me during the internship.

Finally, I would like to thank my family, who supported me during this journey and to whom I dedicate this thesis.

Preface

The major content of all chapters was developed during the PhD. However, the Chapter 3 is partly built on the ideas I explored during my MPhil in Advanced Computer Science. As such there is some overlap between Section 3.3 and the MPhil dissertation. The overlap regards presentation of node2vec (see Subsection 3.3.2) and CCA (see Subsection 3.3.3) models.

Table of contents

List of figures

List of tables

List of tables

1

Introduction

1.1 Motivation

One of the long-standing goals in Natural Language Processing (NLP) has been to represent the 'meaning' of language units such as words, phrases and sentences in a mathematical construct that would allow a computer to understand the intent conveyed by the unit. To pursue this goal, many different schemes to meaning representation have been proposed. In the literature, however, one can distinguish two broad categories: 1) representation of meaning in a symbolic structure, and 2) representation of meaning in a numeric vector. In principle, there are two differences between them. First, is the mathematical object for meaning representation: discrete vs continuous. Second, is the amount of domain-specific[1] information we need to incorporate into the models to represent the meaning of language units; the first category "hard-code" the representation of meaning according to some formal system, while the models from the second category learn the meaning from data with the minimum set of assumptions. This second difference is especially of interest to this thesis.

The common principle of schemes from the first category is to rely on atomic symbols and some form of structure, represented either with formulae or as a graph, that establishes a relation between these symbols. This category includes, but is not limited to methods such as Abstract Meaning Representation (AMR; Banarescu et al. (2013)) and approaches from

[1]Domain is a quite general term and used differently depending on the context. It can refer to: 1) a corpus (e.g. biomedical vs movie or more generally to corpora that we constructed using different heuristics (McCoy et al., 2019)), 2) a task, 3) types of data (text or image), and 4) types of knowledge required (e.g. linguistic knowledge). In this chapter we also use it broadly, mainly referring to 3 and 4. However, in Chapters 3-6 we mainly mean 1 and 2.

formal semantics (FS; Montague (1973); Blackburn and Bos (2005)). These methods are usually interpretable - that is following the predefined rules we can understand what and how meaning has been assigned to a linguistic unit. Also, having the predefined rules allows us to express regularities in terms of how meaning is assigned to a previously unseen language unit which in turn leads to generalisation. Unfortunately, AMR and FS methods are not robust if the language units have unfamiliar symbols e.g. a sentence may have a new word, or if the language unit has a novel meaning which cannot be expressed with existing rules[2] (e.g. polysemy, neologism, etc.). This would require an addition of new rules and symbols, and their incorporation into an existing scheme may be time-consuming and expensive as domain expertise is needed. Despite the ongoing research that aims at improving the understanding of existing schemes and addressing their shortcomings (Abend and Rappoport, 2017) the aforementioned problems lead to the decline of the popularity of these methods.

Instead, methods from the second category have gained a lot of attention.[3] The unifying characteristic of these methods is that meaning of a linguistic unit is distributed in a high-dimensional numeric vector. Where each dimension of the vector represents a certain aspect of the meaning. To derive values of the vector, the approach relies on statistical models and large amounts of text which can be in a raw form, or along with additional attributes to characterise specific semantic properties of text e.g. a label expressing positive or negative sentiment of a sentence. The most successful methods from this category use neural networks as a statistical model and the distributed real-valued vector representations of text (a.k.a. embeddings; Bengio et al. (2003), Mikolov et al. (2013)). In this thesis, we solely concentrate on these models.

Compared to their symbolic counterpart, neural networks eschew any use of symbolic structure. Moreover, they are generic models in a way that they can be applied both to

[2]Language is creative which implies the use of words and semantic/syntactic structure in an "unconventional" way. As such, there is a high chance that there could be a sentence/phrase that has a novel semantic/syntactic structure that are not covered by the existing rules.

[3]We refer to two periods: statistical NLP (1990) and neural NLP. Neural networks became popular in 2013 (Ruder, 2018). However, worth noting that the first neural language model, which learns distributed representation of words, was introduced in 2001/2003 (Bengio et al. (2003) the publication date varies for this paper). We refer a reader to Ruder (2018) for a review of the history of neural networks in natural language processing.

the text and image data without significant change in the architecture of the models.[4] Still, neural models such as BERT (Devlin et al., 2019), which are mainly trained on an unlabeled corpus,[5] achieve state-of-the-art (SOTA) performance on language tasks that are designed to test the competence of a model on understanding the meaning of language unit(s), e.g. GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019). Motivated by these empirical results, the field of NLP has been progressing by scaling up the models and training them on larger corpora. This allowed to achieve even better performance on downstream tasks (Liu et al., 2019b).

Recently, however, the success that these neural models achieved on downstream tasks has been questioned in terms of evaluation techniques (Yogatama et al., 2019; Lazaridou et al., 2021), their abilities to perform reasoning tasks (Marcus, 2020) and their capabilities to capture meaning (Bender and Koller, 2020). Moreover, there have been recent findings that question the robustness of the neural networks to unfamiliar language units in out-of-distribution settings (McCoy et al., 2020b). These findings and discussions indicate that we may need something more than just scaling up domain-agnostic neural network models and data to represent the meaning of language units.

One paradigm that can address many of the mentioned above limitations (e.g. reasoning and robustness) and as such should be considered as a promising direction to bridge the gap between human and machine intelligence (Marcus, 2020) is the integration of the neural and rule-based models from formal semantics, also known as neural-symbolic models.[6] This paradigm has been adopted by many researches (Andreas et al., 2016; Minervini et al., 2018; Yi et al., 2018, 2020) and shapes an active area of research. Recently, though, it has been challenged by Ding et al. (2020). They demonstrated that neural networks, without the use of symbolic programs, can outperform neural-symbolic models on visual-based reasoning tasks (Girdhar and Ramanan, 2020; Yi et al., 2020). Hence, this questions the

[4] For example, Transformer (Vaswani et al., 2017) model which was originally designed to model text data, however, has been recently adapted with minimum changes to images (Dosovitskiy et al., 2021).

[5] Usually, the models are first trained on large unlabelled corpus and then are fine-tuned with a small number of labeled examples.

[6] Here we mainly refer to the models that: 1) use a neural network as a component that produces an output to a symbolic program (Yi et al., 2018) and 2) use neural networks as a soft relaxation of a predefined (not learned) symbolic program (Andreas et al., 2016).

necessity of the symbolic components in the neural networks, at least for visual-based reasoning. Nevertheless, in this thesis: 1) we do not exclude the possibility that a hybrid of symbolic/discrete and continuous variables (Martins, 2021) maybe needed in order to model the meaning of a language unit (Baroni, 2019), 2) we argue not against the neuro-symbolic integration and rather leverage them in parts of our work for injecting *inductive biases* (Mitchell, 1980; Griffiths et al., 2010; Battaglia et al., 2018; Goyal and Bengio, 2021).

Inductive biases is another paradigm that may allow us to alleviate the aforementioned limitations of neural networks.[7] Usually, there are many equally good solutions to a task; an inductive bias is what allows a learning algorithm to prefer one solution over the others. There are many different ways in which an inductive bias can be incorporated into neural networks. Common approaches include but not limited to: 1) including it directly into an architecture of the neural networks e.g. use of convolution and pooling operations (Gholamalinezhad and Khosravi, 2020) in the convolutional neural network networks (CNNs; LeCun et al. (1989), Ciresan et al. (2011)) makes them invariant to translations in the input data, 2) using data augmentation techniques that construct synthetic text out of existing corpus to allow a model generalise better to previously unseen text (Andreas, 2020), and 3) using multitask learning to force a neural network to learn the weights such that it would generalise to multiple tasks (Caruana, 1993).

This poses an interesting question - what inductive biases do we need to model language units and meaning that they convey? There are two main lines of research that try to answer this question. The first line investigates how the existing inductive biases in the neural architectures affect their ability to model the language units (Tran et al., 2018; Ravfogel et al., 2019; McCoy et al., 2020a). The second line tries to incorporate explicit inductive biases of interest into neural networks and then study the effect these biases have on modeling language units (Dyer et al., 2016; Bahdanau et al., 2019; Shen et al., 2019; Ding et al., 2020; Słowik et al., 2020). The ideas presented in this thesis are closer to the second line of research. We

[7]Inductive biases are powerful but as general principles they are highly varied, and have also been successfully employed in other machine learning paradigms: Bayesian models (prior belief; Griffiths et al. (2010)), regularization (Occam's razor), k-neareast neighbours (smoothing; Wagner et al. (2018)), support vector machines (iter-class distance Zhang et al. (2012)), etc.

explore various ways of incorporating inductive biases into distributed text representations, where the granularity of text representations varies from word level to sentence level. There are three main inductive biases that are explored in this thesis:

- relational inductive bias - bias a learning algorithm to be reflective of relationships that exist between language units;
- information-theoretic inductive bias - bias a learning algorithm to be constrained by an information-theoretic notion, encoding channel capacity;
- sparsity inductive bias - bias a learning algorithm to utilise different subspaces of the representation space by inducing sparse representations of the language units;

In the next section, we will elaborate more on these biases.

1.2 Research Questions (RQs)

In what follows we provide an overview of research questions and contributions of this thesis.

1.2.1 RQ 1: Relational Inductive Bias for Words (Data-Based)

Research Questions. To learn reliable word embeddings SOTA models need large amount of text (frequently) containing these words. One of the reasons for this is the lack of inductive biases that would allow the models to select the meaning of a word out of possible alternatives. Hence, words that are not frequent or absent in the text cannot be represented reliably with the embedding. However, can we use an inductive bias that allows us to use much smaller amount of data and still learn a good representation for these rare and unseen words? We hypothesise that if such a bias exists, in order to reduce the amount of data needed to learn the meaning of a word,

it should explicit express (bias) the meaning of the word. Here, we investigate if a Knowledge Graph[8](KG) can be used as such inductive bias.

Use of Inductive Bias. We bias the learning algorithm to derive a semantic representation of a word in terms of relationships that exist between the word and other words in a KG.

Main Contributions. We propose a framework that exploits the semantic structure of the lexical resource for inducing embeddings of unseen words.

Summary. Word embedding techniques heavily rely on the abundance of training data for individual words. Given the Zipfian distribution of words in natural language texts, a large number of words do not usually appear frequently or at all in the training data. In this work we put forward a technique that exploits the knowledge encoded in lexical resources, such as WordNet, to induce embeddings for rare and unseen words. Our approach adapts graph embedding and cross-lingual vector space transformation techniques in order to merge lexical knowledge encoded in KGs with that derived from corpus statistics. We show that the approach can provide consistent performance improvements across multiple evaluation benchmarks: intrinsic, on multiple rare word similarity datasets, and extrinsic, in two downstream text classification tasks.

1.2.2 RQ 2: Relational Inductive Bias for Sentences (Data-Based)

Research Questions. Learning semantic representation of sentences/phrases is an immensely hard task because numerous possible meanings can be expressed by composing the words in the sentences/phrases. One way to alleviate this issue is via a supervision signal that expresses (or biases) the meaning of the sentences. However, what would be the ‘right‘ supervision signal to learn the meaning of sentences? We

[8]Because of confusion between Knowledge Bases, Ontologies and Knowledge Graphs (KGs) terminology (Ehrlinger and Wöß, 2016), KGs are used here as a general term for representing *knowledge* in the form of a graph.

follow the work of Hill et al. (2015a) who proposed to use dictionary definitions to learn the meaning of sentences. We extend this idea and instead of mapping a dictionary definition to a single word we map it to path graphs extracted from a KG. As a first step towards investigating whether this is a right bias to learning meaning of sentences we pose the following question: can we bias semantic representation of a sentence to be reflective of topological dependencies that exist in a KG?

Use of Inductive Bias. We bias the learning algorithm to restrict the semantic representation of sentences in terms of relationships that exist between the entities in a KG.

Main Contributions. We present a framework that allows to learn semantic representation of sentences in terms of topological dependencies that exist between entities of a KG.

Summary. We present a novel method for mapping unrestricted text to knowledge graph entities by framing the task as a sequence-to-sequence problem. Specifically, given the encoded state of an input text, our decoder directly predicts paths in the knowledge graph, starting from the root and ending at the target node following hypernym-hyponym relationships. In this way, and in contrast to other text-to-entity mapping systems, our model outputs hierarchically structured predictions that are fully interpretable in the context of the underlying KG, in an end-to-end manner. We present a proof-of-concept experiment with encouraging results, comparable to those of state-of-the-art systems, indicating that sentence embeddings do incorporate the semantics of the path graph. In this case, the semantics is the hypernymy hierarchy of concepts.

1.2.3 RQ 3: Information-Theoretic Inductive Bias for Sentences (Data-Agnostic)

Research Questions. Taking the amount of information that is encoded about a sentence in its embedding as a form of inductive bias, what would be the implications of regulating this?

Use of Inductive Bias. We control the amount of information about a sentence that is encoded in its sentence embedding.

Main Contributions. Autoencoders are popular for unsupervised representation learning. In principle, autoencoders try to preserve as much as possible information about the data they model. However, it is yet poorly understood how much information should be preserved. In this thesis, we study a variant of Variational Autoencoder (VAE) model that allows us to explicitly control the amount of information that is encoded in a sentence embedding. We treat it as a form of inductive bias that the model uses to learn sentence embeddings. We further analyse the effect of this bias on the quality of the learned sentence representations on two downstream tasks: text generation and text classification.

Summary. We explore the effect of the amount of mutual information between a sentence and its representation on downstream tasks. For this we use VAE framework. VAEs are known to learn rich representation of text. Besides, thanks to the encoder-decoder architecture we can study the goodness of learned sentence embeddings on discriminative tasks as well as text generation. Part of its success is attributed to the Kullback-Leibler (KL) divergence term inside the VAE objective function. We impose an explicit constraint on the KL term to understand its significance in controlling the amount of information about a sentence is transmitted to its representation. Within this framework, we explore different properties of the estimated posterior distribution, and highlight the trade-off between the amount of information encoded in a sentence representation during training, and the generative

behaviour of the model. Furthermore, we analyse the discriminative performance of learned representations on three text classification tasks.

1.2.4 RQ 4: Sparsity Inductive Bias for Sentences (Data-Agnostic)

Research Questions. Can sparse sentence embeddings, learned with sparsity inductive bias, match (or outperform) the performance of their dense counterpart on downstream tasks? What are the necessary conditions for this to happen?

Use of Inductive Bias. We bias a learning algorithm to utilise separate subspaces of the representation space.

Main Contributions. We present a novel VAE model - Hierarchical Sparse Variational Autoencoder (HSVAE) that allows to induce sparse representations of large units of text. Also, using HSVAE as a testbed, we establish how statistical properties of a corpus such as word distribution in a class affect the ability of learned sparse codes to represent task-related information, and show its impact on text classification tasks.

Summary. It has been long known that sparsity is an effective inductive bias for learning efficient representation of data in vectors with *fixed dimensionality*, and it has been explored in many areas of representation learning. Of particular interest to this work is the investigation of the sparsity within the VAE framework which has been explored a lot in the image domain, but has been lacking even a basic level of exploration in NLP. Additionally, NLP is also lagging behind in terms of learning sparse representations of large units of text e.g., sentences. We use the VAEs that induce sparse latent representations of large units of text to address the aforementioned shortcomings. First, we move in this direction by measuring the success of unsupervised state-of-the-art (SOTA) and other strong VAE-based sparsification baselines for text and propose a hierarchical sparse VAE model to address the stability issue of SOTA. Then, we look at the implications of sparsity on text classification across 3 datasets, and

highlight a link between performance of sparse latent representations on the downstream tasks and its ability to encode task-related information.

1.3 List of Publications

This thesis is based on the following works:

- **Victor Prokhorov**, Mohammad Taher Pilehvar, Dimitri Kartsaklis, Pietro Lio, Nigel Collier (2019).“Unseen Word Representation by Aligning Heterogeneous Lexical Semantic Space” In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Honolulu, Hawai: vol 33(01), pp.6900-6907. URL: https://doi.org/10.1609/aaai.v33i01.33016900 **(RQ 1: see Subsection 1.2.1)**

- **Victor Prokhorov**, Mohammad Taher Pilehvar and Nigel Collier (2019).“Generating Knowledge Graph Paths from Textual Definitions using Sequence-to-Sequence Model” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Minneapolis, Minnesota: Association for Computational Linguistics, pp. 1968–1976. URL: https://www.aclweb.org/anthology/N19-1196 **(RQ 2: see Subsection 1.2.2)**

- **Victor Prokhorov**, Ehsan Shareghi, Yingzhen Li, Mohammad Taher Pilehvar and Nigel Collier (2019). “On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation” In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Hong Kong: Association for Computational Linguistics, pp.118–127. URL: https://www.aclweb.org/anthology/D19-5612 **(RQ 3: see Subsection 1.2.3)**

- **Victor Prokhorov**, Yingzhen Li, Ehsan Shareghi and Nigel Collier (2021) “Learning Sparse Sentence Encoding without Supervision: An Exploration of Sparsity in Variational Autoencoders” In: *Proceedings of the 6th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics. URL: https://aclanthology.org/2021.repl4nlp-1.5/ **(RQ 4: see Subsection 1.2.4)**

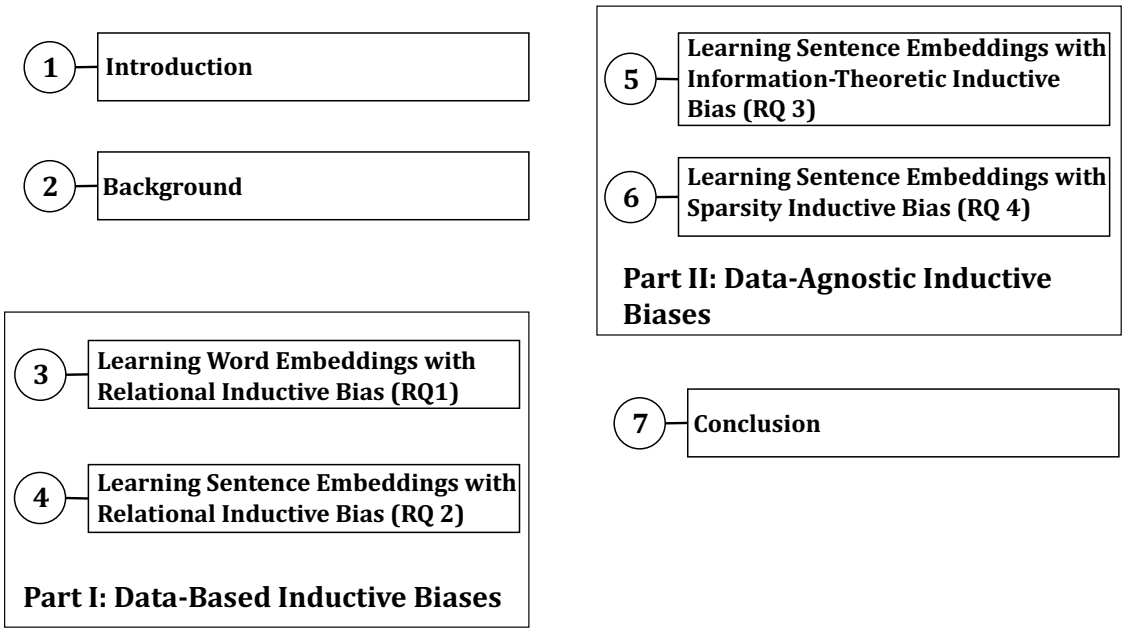


Fig. 1.1 Roadmap. RQ stands for Research Question (see Section 1.2)

Works that are not included in this thesis:

- Mohammad Taher Pilehvar, Dimitri Kartsaklis, **Victor Prokhorov** and Nigel Collier (2019). "Card-660: A reliable evaluation framework for rare word representation models" In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1391–1401. URL: https://www.aclweb.org/anthology/D18-1169

- Lan Zhang, **Victor Prokhorov** and Ehsan Shareghi (2021) "Unsupervised Representation Disentanglement of Text: An Evaluation on Synthetic Datasets" In: *Proceedings of the 6th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics. URL: https://aclanthology.org/2021.repl4nlp-1.14/

1.4 Thesis Outline

The rest of the thesis is structured as follows:

Chapter 2. In this chapter, we introduce the relevant NLP and Machine Learning (ML) concepts. This would make this thesis self-contained and allow a reader to better understand the work presented in the content chapters: Chapters 3, 4, 5 and 6.

Chapter 3. This chapter builds on RQ 1 (see Subsection 1.2.1). We study if the relational semantic constraints that exist between the entities in an ontology serve as a good inductive bias to learn word representations for rare or unseen words. To demonstrate this, our approach adapts graph embedding and cross-lingual vector space transformation techniques. We show that the approach can provide consistent performance improvements across multiple evaluation benchmarks: on multiple rare word similarity datasets and in two downstream text classification tasks.

Chapter 4. This chapter builds on RQ 2 (see Subsection 1.2.2). We investigate whether semantic relationships that exist between the entities in an ontology can be incorporated into larger units of text such as sentences. For this, we present a novel method for mapping dictionary definitions to knowledge graph entities by framing the task as a sequence-to-sequence problem. In this work, we only perform an intrinsic evaluation of the model, by demonstrating that sentence embeddings do incorporate the semantic relationships between the entities. Evaluation of the effectiveness of this inductive bias on other downstream tasks is left for further work.

Chapter 5. This chapter builds on RQ 3 (see Subsection 1.2.3). We concentrate on an aspect of the representation learning process via VAEs which is motivated from information-theoretic perspective: the amount of information the latent sentence representation stores about a sentence. We conduct a set of qualitative and quantitative experiments to demonstrate how this quantity affects the generative capacity of VAEs as well as the discriminative performance of latent representations on three text classification tasks.

Chapter 6. This chapter builds on RQ 4 (see Subsection 1.2.4). We propose a novel framework, Hierarchical Sparse Variational Autoencoder (HSVAE), that imposes sparsity

on sentence representations via direct optimisation of Evidence Lower Bound (ELBO). Our experimental results illustrate that HSVAE is flexible and adapts nicely to the underlying characteristics of the corpus which is reflected by the level of sparsity and its distributional patterns. Thus, we highlight a link between performance of sparse latent representations on three text classification tasks and its ability to encode task-related information. Also, using the text classification corpora as a testbed, we established how statistical properties of a corpus such as word distribution in a class affect the ability of learned sparse codes to represent task-related information.

Chapter 7. We summarise the work that is presented in the content chapters and also discuss potential future directions based on the findings of the thesis.

2

Background

The works we present in this thesis are built on two core concepts: *distributed representations* of language units, and *inductive biases*, which are in turn discussed within the context of *neural networks*. In this chapter, we introduce these principles. This serves two purposes. First, by providing a reader with the necessary foundations, we make this thesis self-contained. Second, we also ground our work in the existing literature and give a broader perspective on the research we present here. At the same time, we neither provide an exhaustive literature review nor, unless necessary, formally state existing algorithms. We start with an introduction of neural networks in Section 2.1 then we talk about distributed representation of text in Section 2.2 and finally we put forward a discussion about inductive biases in NLP in Section 2.3.

2.1 A Brief Introduction to Neural Networks

In this section, we aim to briefly introduce neural networks. As a result, we give a crude introduction to this rich topic; quite often omitting the formal definitions of the concepts. We familiarise the reader with all the main concepts applied in this thesis. We refer the reader to Goodfellow et al. (2016), for a more detailed introduction.

2.1.1 Neural Network's Concepts

A neural network is a *differentiable*, *nonlinear*, *parametric* function $f(\cdot;\theta)$, with the parameter θ, that relates two types of variables: x with y, commonly know as input and output

variables, respectively. The mapping of x into y is performed by a series of affine transformations followed by dimension-wise nonlinear mappings. A neural network can comprise of several layers (sub-neural networks): $f(\cdot;\theta) = f_3(f_2((f_1(x;\theta_1));\theta_2);\theta_3)$, where $f_i(\cdot;\theta_i)$ is also a neural network. These intermediate layers (layers that are not an input layer[1] and do not directly map to output) are also known as *hidden layers* because we do not have explicit variables into which they should map domain points.[2]

Depending on the type of the input and output variables and how one processes them the form of the function, $f(\cdot;\theta)$, varies (see Subsection 2.1.2). For example, x can be a simple vector and y be a scalar value in which case a feedword neural network can be used. However, one may have more complex objects for both input and output variables, i.e. x can be a sentence and y be a syntax tree, in which case one can use an encoder-decoder neural network, with both encoder and decoder suitable to process these variables. In principle though, whether to use a data specific neural network is often a matter of efficiency.[3] Furthermore, neural networks are universal function approximators (Cybenko, 1989; Hornik, 1991). That is, given large enough number of parameters, one can always represent an association between the input and output variable, up to some error.

With a variable degree of "correctness", there are many possible functions that allow us to relate the two variables. However, one needs to choose a function, $f(\cdot;\theta)$, out of the possible alternatives, $\{f(\cdot;\theta)|\theta \in \Theta\}$, where Θ is the set of possible parameters θ the function can have, that results in the best association. A common way to achieve this is to use an algorithm (see Subsection 2.1.3) that adjusts the parameter θ to minimise an *error function* (or objective function). This error function estimates the fitness of the mapping from x to y, given the 'true' y. If the error is "small enough" then the function is thought to associate the two variables correctly. This process, in the literature, is known as *training* of neural networks (see Subsection 2.1.3).

[1] An input layer is simply an input variable x.

[2] Here we refer to the domain as a set which comprises of inputs of a function.

[3] Some neural network were specifically designed to process data with certain properties. For example, recurrent neural network (Rumelhart et al., 1988) were designed to process sequential data such as sentences.

To train a neural network one uses a *training dataset* (or corpus), which is a set of (x, y) pairs.[4] During the training, to judge how well the neural network learned the association between the input and output variables a *validation dataset* is used. It, however, is not used to train the model but instead allows us to understand if the neural network is *overfitting*[5] to the training examples. After the neural network is trained it is further evaluated on the *test dataset*. Test dataset, also, is not used during the traning of the neural network and instead allows us to compare performance of various neural architectures that are proposed to associate the variables.

2.1.2 Types of Neural Networks

In this section we introduce five main types of neural networks: 1) feedforward neural networks, 2) recurrent neural networks, 3) neural networks with input-dependent structure, 4) encoder-decoder neural networks, and 5) attention-based neural networks. There are, potentially, many ways to group the existing architectures into meaningful taxonomies. Our classification scheme is based on the flow of information (that is a process of how an input variable is being mapped into output variable) in the neural networks and it also covers the types of neural architectures we use in this thesis. Furthermore, this classification scheme allows us to abstract from the specifics of a particular architecture and instead focus on high-level properties of the neural networks.

Feedforward Neural Networks. As the name suggests, the information flows in the forward direction from an input variable x until it reaches the output variable y. For example, in a feedforward neural network with three layers,[6] the flow of information can be represented as a composition of functions (layers) i.e. $y = f_3(f_2((f_1(x;\theta_1));\theta_2);\theta_3)$. There are two distinguishing characteristics of feedforward neural networks: 1) $f_i(\cdot;\theta_i)$ can not reuse its output values, in other words, it does not have recurrent connections, and 2) there are no

[4] In unsupervised learning (see Subsection 2.1.3), a dataset only comprises of x.

[5] A neural network may fit the training examples too closely to an extent that it may become useless to predict any future associations between the pair of variables (x, y).

[6] Excluding the input layer

loops, i.e. information can not flow from $f_{i+1}(\cdot;\theta_i)$ back to $f_i(\cdot;\theta_i)$. The functional form of $f_i(\cdot;\theta_i)$ depends on the data (x,y) one wants to process. Two archetypal examples of these networks are a multilayer perceptron and a convolutional neural network (Lecun et al., 1998).

Recurrent Neural Networks. Given a sequence $\{x_t\}_{t=1}^N$ of N symbols, recurrent neural network (RNN; Rumelhart et al. (1988)) allows the information to flow from a symbol x_t, where t is the position of the symbol in the sequence, to all the consecutive symbols $x_{t+1}, x_{t+2}, .., x_N$. To achieve this, it uses the following parametric function: $h_t = f(h_{t-1}, x_t; \theta)$, where h_t is the vector representation of the sequence $x_t, x_{t-1}, .., x_1$. In other words, RNN uses both its previous output h_{t-1} and the current input x_t to process information. Also note, in this case, there is either an output variable y_t for each symbol x_t or only one output variable y for x_N. Two of the most common variants of RNN are Long Short-Term Memory Network (LSTM; Hochreiter and Schmidhuber (1997)) and Gated Recurrent Neural Network (GRU; Cho et al. (2014a)). Similar to the feedforward neural networks, one can also compose several RNNs together. The way the RNNs are composed is task dependent, but the most common way is to feed the sequence $\{h_t^i\}_{t=1}^N$ of length N, produced by $f_i(h_{t-1}^i, x_t; \theta)$, to $f_{i+1}(h_{t-1}^{i+1}, h_t^i; \theta)$.

Neural Networks with Input-Dependent Structure. Neural network with the input-dependent structure, rely on a predefined symbolic structure (the structure varies with the input), i.e. graph, to explicitly define information flow. For example, with Recursive Neural Networks (Pollack, 1990) one can combine word embeddings into sentence embeddings according to a predefined tree structure (Socher et al., 2011). To use more generic graphs, one can use Graph Neural Networks (Scarselli et al., 2009; Kipf and Welling, 2017). In this case, we can have an output variable y for the whole graph or for each node of the graph. Also, worth noting that the RNNs and CNNs are also structured neural networks. However, the structure is the same for all the inputs. For example, if in Recursive Neural Networks we fix the structure to a linear chain then they will be equivalent to RNN.[7] Also, in CNN we define structure with the convolutional filters.

[7] One remark, depending on the number of words in a sentence, the number of nodes in the chain will vary.

Encoder-Decoder Neural Networks. With the encoder-decoder neural network the information flows as follows: First, an input variable is encoded into a hidden representation h and then, the output variable is decoded while being conditioned on the hidden representation. The first step is done with an encoder network and the second step with a decoder network. A functional form of a parametric function both in encoder and decoder can be any of the aforementioned neural networks. A special case of the encoder-decoder neural network is called *autoencoder*. The distinguishing characteristic of autoencoders is that the output variable is the same as an input variable, i.e $y = x$.

Attention Based Neural Networks. Attention-based neural networks were first introduced in the context of machine translation (Bahdanau et al., 2014), where the neural network is an encoder-decoder, with an RNN encoder and decoder. At each decoding step, the attention mechanism allows selecting the most suitable h_t produced by the encoder. In other words, it facilitates a dynamic flow of information from the encoder to the decoder. The attention mechanism is not only being used during the decoding, it can also be used in the encoding of sequence. For example, in the encoder, one can use *self-attention* mechanism (Vaswani et al., 2017) to learn context based representation h_t, where suitable context is selected via the attention mechanism. This, also, allows to dynamically select a suitable context for each input variable x. Furthermore, recent works try to establish the connection between the attention based models e.g. Transformers (Vaswani et al., 2017) and the structured neural networks (Liu et al., 2019a; Joshi, 2020).

2.1.3 Training of Neural Networks

To train the aforementioned neural networks one, most often, uses gradient-based learning algorithms (Duchi et al., 2011; Kingma and Ba, 2015; Dozat, 2016; Bottou et al., 2018). The gradient-based learning comprises of three steps. The first step is forward propagation. At this step the information propagates from x through a neural network until it reaches the output variable y. During the forward propagation, we get an estimate $\hat{y}$ of y and calculate how close the estimate $\hat{y}$ is to the true value of y via an error function. The second step is the

back-propagation (Rumelhart et al., 1986) computing a gradient (∇_θ) of the error function ($L(y,\hat{y})$) with respect to parameter θ of a neural network. Finally, once we calculated the gradient we can use one of the gradient-based learning algorithms to update the value of the θ. One of the most popular techniques is Stochastic Gradient Descent (SGD; (Kiefer and Wolfowitz, 1952; Robbins, 2007; Bottou et al., 2018)) which updates the weights as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta L(y,\hat{y}),$$

where β is a learning rate.

Depending on the availability of output variable y one can distinguish two types of learning: *supervised learning* and *unsupervised learning*. We discuss these two types of learning below:

Supervised Learning. In supervised learning, the output variable y is available and also differs from the input variable x. In NLP, it is usually expressing some explicit form of semantics, e.g. in a dataset we can have a sentence as an input variable x and its sentiment, represented as an integer number, as an output variable y. Also, y can be a desired output that neural network should produce once it processed the input variable x. For example, x can be a sentence and y can be its syntactic parse tree. The variable y, in supervised learning, is also commonly referred to as a label.

Unsupervised Learning. In unsupervised learning, the output variable y is not available or a version of the input variable x. This type of learning is most commonly used to train autoencoders, where it learns to represent x in some latent representation h produced by the encoder. Also, closely related to unsupervised learning is *self-supervised learning*, where we modify the input variable x, according to some heuristics and use the modified version as an output variable y. For example, we can remove some words from a sentence x and make the neural network predict these words (Devlin et al., 2019).

2.2 Distributed Text Representations

A distributed representation instantiates a meaning of a discrete language unit[8] in a continuous vector (embedding), where the meaning is distributed along each dimension. A collection of such embeddings form a semantic space where semantically similar language units are of closer proximity to each other than language units that are semantically distinct. In this thesis, we assume that the meaning can be learned either from a distribution of words/sentences in a raw text (the distributional hypothesis (Harris, 1954; Firth, 1957), which equates meaning of a word to the context in which this word occurs; thus words that occur in a similar context tend to have similar meaning) or (and) from a supervised signal.

2.2.1 Word Embeddings

In modern NLP, the most prominent models that have been used to learn the word embeddings–real-valued vector representations of words–are neural networks (Bengio et al., 2003; Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2019). These models vary in design, but their unifying characteristic is the reliance on a form of word prediction given a context. This approach to learning is inspired by the distributional hypothesis.

Earlier approaches focus on learning *static word embeddings* (Bengio et al., 2003; Mikolov et al., 2013). That is after training of the model their representation stays the same;[9] given a symbolic representation of a word one can always retrieve the same vector representation. In other words, we have a deterministic function $e : \mathbb{W} \rightarrow \mathbb{R}^d$ that maps a symbolic representation of a word $\in \mathbb{W}$ into a d-dimensional vector $\mathbb{R}^d$, e.g. $e(\text{'cat'}) \rightarrow (a_1, a_2, ..., a_d)$, where $a_i \in \mathbb{R}$. However, this is somewhat a sub-optimal approach, because a word can have multiple related (polysemy) or unrelated (homonymy) meanings and its meaning often depends on the context in which it occurs.

One approach to remedy this issue is to learn multiple embeddings per word - where each embedding represents a sense of that word (Reisinger and Mooney, 2010; Neelakantan

[8] In this thesis we only consider written language.

[9] The word embeddings can be further updated with a supervision signal if reused on downstream, but otherwise their representation is constant.

et al., 2014). An apparent limitation of this approach is that we need to store multiple vectors per-word instead of one. Another approach proposes to learn a function that maps a symbolic word to a continuous vector depending on the context of this word. This removes the need to store multiple sense embeddings for each word. That is, in an abstract form, there is a function $e : \mathbb{W} \times \mathbb{W}^N \times \Theta \rightarrow \mathbb{R}^d$ that takes as an input a word $\in \mathbb{W}$, its context $\in \mathbb{W}^N$ of N words, and parameters Θ (which is represented by a neural network) and outputs an embedding of this word. When the context in which the word occurs changes, so does its representation. This type of word embeddings is commonly known as *contextualised word embeddings* (McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019).

Modelling the meaning of smaller language units such as words with embeddings has had great success. It has been shown that learning such embeddings on large unlabelled corpora and reusing them in a model that solves a downstream task, usually, improves the performance (Turian et al., 2010; Peters et al., 2018; Devlin et al., 2019). This approach of training a neural network model on one task and then repurposing its parameters (or a subset) for the other task in machine learning literature is known as *transfer learning*. This allows better generalisation to various language task. However, how to model word embeddings is still an open question. To illustrate this we discuss below the two most prominent pitfalls that the aforementioned models have:

The amount of text and size of the models (Pitfall 1): Current neural models comprise of millions and sometimes billions of parameters and are trained on billions of words. Their training is very expensive and time consuming (Sharir et al., 2020). Moreover, as it was discussed in Lazaridou et al. (2021) simply scaling the size of neural networks cannot solve the problem such as temporal generalization - ability of a model to adapt to continuous change of language use, i.e. use of novel words and novel information generated in ever-changing world. This raises a reasonable question: Whether we need to rethink our approach to the model design and devote more time to incorporate inductive biases that allow the expected generalisations.

Modelling rare and unseen words (Pitfall 2): Quality of a learned word embedding depends on the frequency of its appearing in a corpus. As such, these models cannot learn a good representation when a word is not (frequently) present in a training corpus, which can happen due to Zipfian distribution of words in a natural text. One way to address this problem is to segment a word into characters or subwords - a substring of characters that can be smaller than a word, e.g. BPE (Sennrich et al., 2016b). These approaches have been tried both for static word embeddings (Bojanowski et al., 2017) and contextualised word embeddings (Peters et al., 2018; Devlin et al., 2019). While with this approach one can represent a previously unseen word, as the recent study shows (Lazaridou et al., 2021), the quality of embeddings induced for rare and unseen words is still lagging behind. We address this pitfall in Chapter 3.

2.2.2 Sentence Embeddings

In a natural language, larger units are constructed from smaller ones following certain 'rules' of that language. For example, to construct a sentence we combine individual words together according to *syntactic rules* of that language. Furthermore, when constructing multiple larger units that convey different meaning, smaller units are, most often, reused. For instance, in the following two sentences: 1) *A dog chases a cat* and 2) *A cat chases a dog*, the words that we use are the same, but these two sentences convey different meanings. In other words, natural language is *productive*. That is, in case of sentence construction, we can generate, potentially unlimited number of sentences conveying novel meanings reusing a finite set of words.

As such, to extend the distributed semantics to phrases and sentences, referred to as *compositional distributed semantics*, one needs to propose a compositional operator (function) to combine word embeddings into a sentence embedding preserving the aforementioned properties of a natural language. One possibility is to use a basic arithmetic operator, e.g. addition '+'. However, this has an obvious drawback, the addition of two vectors is invariant to order. As such, with '+' we can not distinguish the meanings of the two mentioned English sentences as they would have identical sentence embeddings. Surprisingly, despite

failing to model some properties of a natural language, this approach[10] is shown to be a strong baseline for many downstream tasks (Blacoe and Lapata, 2012; Arora et al., 2017; Reimers and Gurevych, 2019).

Alternatively, since we do not know the right way to combine words, we can create a model with a minimum set of assumptions and let it learn how to compose the words into sentences by training it on a corpus. In contemporary NLP, this model is typically a neural network. Despite having a more complex compositional capacity, they have also been criticised (Baroni, 2019). One of the most ubiquitous criticism is that they do not reuse previously learned compositional 'rules'. For example, if a neural model learned that *jump twice* means *JUMP JUMP* it may fail to infer that *dax twice* means *DAX DAX*. Additionally, a neural network's compositional capacity depends on its architecture and, most importantly, one's formalisation of the principle of compositionality (Janssen, 1997).

Scaling the distributed semantics to larger language units is still an open problem. In general, research in compositional distributed semantics has two main priorities (Hill et al., 2016). The first priority is the design of architecture of neural networks (Elman, 1990; Socher et al., 2010; Kalchbrenner et al., 2014; Vaswani et al., 2017), such that they would better model properties of a natural language. For example, Socher et al. (2011) proposed to use recursive autoencoders to exploit the hierarchical structure of the sentences. Also, Kalchbrenner et al. (2014) accentuate their attention on the encoder and propose to use Dynamic Convolutional Neural Network to better model semantics of sentences.

The second priority is the design of a task or/and objective function, either in supervised or unsupervised settings, to train the neural networks that would allow them to represent semantics of an arbitrary language unit. In the supervised setting, Stanford Natural Language Inference (Bowman et al., 2015a) is the most common task that is used to improve the semantic representation of sentence embeddings (Conneau et al., 2017; Cer et al., 2018; Reimers and Gurevych, 2019). Though, Subramanian et al. (2018b) argues that training a model on multiple tasks allows it to generalise better. In the unsupervised setting, the most

[10]Here, we only talk about '+', but the majority of the works use averaging of word embeddings $s = \frac{1}{N}\sum_i^N w_i$. Since, the averaging also violate the properties of a natural language we discuss above, in this brief discussion we do not differentiate between the two compositional operators.

prominent model was proposed by Kiros et al. (2015), where they use a similar strategy as skip-gram word2vec model (Mikolov et al., 2013) by encoding a sentence and predicting its adjacent sentences. Nowadays, it is more common to use BERT model (Devlin et al., 2019)[11] to represent a sentence embedding, either by summation/averaging of token embeddings or just with the embedding of the special token '[CLS]'.

To summarise, in order to extend the success of the distributed representations to phrases and sentences we still need to get more definite answers to the following questions:

- What is a preferable neural network architecture to compose words into the larger grammatical units?
- What are preferable distributed representations of the grammatical units - is it a vector or is it a matrix, should it be dense or sparse, should the dimensions of the vector/matrix be structured somehow?
- What task should such a network be trained on?

We address question two in Chapters 5 and 6, and question three in Chapter 4.

2.3 Inductive Biases in NLP Models

There have been multiple works that formulate an inductive bias within a particular machine learning framework (Mitchell, 1980; Caruana, 1993; Griffiths et al., 2010; Battaglia et al., 2018) or overview it for broader machine learning techniques (Goyal and Bengio, 2021). In order not to repeat what has already been said, after introducing some key terminology (see Subsections 2.3.1, 2.3.2, and 2.3.3), we tailor the discussion of the inductive bias to NLP only. To the best of our knowledge, no one has yet made an overview of inductive biases that are used in neural natural language processing systems. Here we initiate this discussion (see Subsections 2.3.4 - 2.3.8)

[11] Also, there have been recent improvement of BERT based sentence embeddings (Li et al., 2020a; Su et al., 2021) that are learned in the unsupervised setting.

2.3.1 What is an Inductive Bias?

To train a neural network we use a training dataset - a set of data points (x, y), see Subsection 2.1.3. Usually, the training dataset is limited in the sense that we are not given every possible x and y pair. This can be due to several reasons. One reason is that it may be impossible to collect all possible pairs of x and y. Another reason can be the cost[12] of collecting the pairs. As such, from the limited number of data points that are available in a training dataset we want a neural network to learn a mapping between the x and y variables that would also hold for unseen (x, y) pairs - this is also known as *induction* (Hume, 1978).

One problem with learning a relation via induction is that given a training dataset there are can be multiple relations that are equally good (Goodman, 1955). However, these relations may not relate previously unseen (x, y) pairs equally well. So, how shall we select a relation out of equally good alternatives? One way of doing this is to use a bias - a principle to prefer one relation over the other. This is known as an *inductive bias* (Mitchell, 1980; Caruana, 1993), which we discuss in this section. There are exist two main definitions of inductive bias:

1. *"bias refers to any basis for choosing one generalization over another, other than strict consistency with the observed training instances"* - Mitchell (1980).

2. *"bias is anything that causes an inductive learner to prefer some hypotheses over others"* - Caruana (1993).

We employ the second definition as it is much broader[13] and allows us to cover the types of inductive biases that we discuss in this thesis.

2.3.2 Types of Inductive Biases

In this thesis, we distinguish two types of inductive biases: 1) data-agnostic (Mitchell, 1980) and 2) data-based (Caruana, 1993). They differ in how they incorporate the bias into the learning process:

[12] It can be either too expensive or time-consuming or sometimes both to collect the pairs.
[13] The second definition subsumes the first.

- Data-agnostic inductive biases make a learning algorithm to prioritise one solution (hypothesis/mapping) over the other independently of training data. This means that we can introduce an inductive bias only to the learning algorithm itself. For example, using l_1 regularisation in the parameters of a neural network or attention mechanism (Bahdanau et al., 2014).

- Data-based inductive biases are based on the assumption that inductive biases can be incorporated into a neural network via a supervision signal presented in training data. It includes approaches ranging from multi-task learning to training data augmentation algorithms.

2.3.3 The Necessity and Danger of Inductive Biases

As argued by Mitchell (1980), bias-free learning is impossible.[14] To better understand this argument lets consider the following scenario.[15] Assume we train our model on a sentiment classification task, where x is a sentence and $y \in \{0, 1\}$ is sentiment label, with 0 indicating a negative sentiment and 1 is positive sentiment. Furthermore, let the training dataset comprises of the following two pairs: (x_1: *I liked this movie*, y_1: 1) and (x_2: *This was a horrible movie*, y_2: 0). A learning algorithm that tries to distinguish a positive movie review from a negative just from these two examples may consider the following two hypotheses:

- h_1 : *classify a review as positive if it has a positive adjective i.e. like, love, etc and negative if it has negative adjectives i.e. horrible, bad, etc.*

- h_2 : *classify a review as positive if it is shorter than a certain length (4 or shorter in this case) and negative if it is longer than a certain length (5 or longer in this case).*

Under the observed training examples these two hypotheses[16] are equally possible. However, it is clear that we would prefer the model to use h_1 in order to classify the

[14] Furthermore, according to the *No Free Lunch Theorem* (Wolpert, 1996; Wolpert and Macready, 1997), it is impossible to design inductive biases that would suit all the tasks.

[15] This example was greatly inspired by the following work Gordon and desJardins (1995).

[16] For h_1 the model needs to know a prior meaning of positive and negative adjectives in order to extend to the examples given above. However, for the sake of this example lets assume that the model can do this.

previously unseen examples. As it is likely that we encounter the positive review that are longer than four words as well as negative reviews that are shorter than five words. Which of the hypotheses the model chooses during the learning process will depend on the inductive biases. That is if we incorporate the 'wrong' inductive bias we may end up with the second hypothesis h_2 and as a result, have a detrimental performance trying to classify the previously unseen reviews. Also, the inductive biases reduce the initial space of possible hypotheses and biasing the model 'too much' can eliminate the hypotheses that are useful for the task.

2.3.4 The Need of Inductive Biases in Neural Language Models[17]

Judging from the reported performance on tasks that require competence in understanding the meaning of language expressions (Wang et al., 2018, 2019) alone, one may conclude that neural models are very close, sometimes, superior to humans (Devlin et al., 2019; Niven and Kao, 2019) in terms of understanding of the language. Partly, such performance can be explained by the ability of the models to acquire certain aspects of linguistic knowledge, as it has been demonstrated with various probing techniques (Belinkov and Glass, 2018; Alishahi et al., 2019; Clark et al., 2019; Coenen et al., 2019; Zhang et al., 2020b). However, to understand if we have the 'right' neural architectures to model a natural language we need to answer the following questions: 1) to what extent the task at hand allow us to make claims regarding the linguistic competence of the model, 2) whether the performance, on the tasks, is achieved due to utilisation of the linguistic knowledge that the models acquire, and 3) how efficient our models are in terms of their size and amount of training data that is needed to acquire the knowledge to reach the performance on the task. To answer these questions we need to look at how these models are being evaluated and understand how efficient they are.

[17]Here, we focus only on neural language models as they have been the main driving force of recent progress in MLP. However, many of the arguments presented here also apply to other neural NLP models. For example, Jia and Liang (2017) used a question answering task to demonstrate that the neural machine comprehension models are not capable to differentiate between a sentence that has a correct answer and a sentence that has a wrong answer but related words with the question.

The Problem of Evaluation and Spurious Patterns in Neural Language Models:

Most of the neural language models are evaluated on a test dataset that is closely related (coming from the same distribution) to the training dataset. However, this may not be enough to test their understanding of the language, as it may simply exploit spurious patterns[18] presented in the training dataset (Niven and Kao, 2019; McCoy et al., 2020b) instead of capturing the "meaning". Indeed, recent works designed more sophisticated evaluation protocols[19] that demonstrate the competence of the models in understanding natural language is far from human.

For example, Yogatama et al. (2019) argue that the models are merely tested to solve a particular dataset rather than a task itself. That is a model may perform well on the SQuAD (Rajpurkar et al., 2016) question answering dataset but if tested again (without further fine-tuning) on the same task but different dataset it is unlikely that it will demonstrate any significant performance. A similar observation was made by McCoy et al. (2020b), where they refer to this phenomenon as inability of the model to generalise to the out-of-distibution (OOD) datasets.[20] As such the evaluation techniques should be designed to test a model on a task rather than a dataset to make a stronger claim about the competence of the model to capture the meaning.

Lazaridou et al. (2021) stress the importance of testing the temporal generalization of neural language models. This is the ability of the models to perform robustly when evaluated on a test dataset that is coming from a different time period[21] e.g. News article for the training dataset is from 1998-2000 period while the test dataset cover the 2005-2007 period. They found that the current state-of-the-art models perform poorly on this test.

[18] Lovering et al. (2021) and Warstadt et al. (2020) explain why a pretrained language model, on a downstream task, may prefer to exploit spurious patterns instead of the linguistic knowledge that it acquired.

[19] Alternatively, conducting an empirical study, Bender and Koller (2020) provide a philosophical discussion where they elaborate why neural language models, which are pretrained only on the language modeling task, are unlikely to capture the meaning of language units but rather artifacts (patterns) presented on a training dataset.

[20] Note, there are works that claim robustness of some neural language models (e.g. BERT (Devlin et al., 2019)) to OOD datasets. For example, (Hendrycks et al., 2020) show that the models are robust to OOD when they are pretrained on many diverse corpora. However, one may argue that because of the size of the training data it may already contain many language expressions that are similar to one in OOD task.

[21] Most of the neural language models are evaluated on a test dataset that has overlapping time periods with the training dataset.

Another line of work uses adversarial examples to demonstrate that the models use spurious patterns available in the data rather than capturing the meaning of a language unit. For example, Niven and Kao (2019) demonstrate, on the argument reasoning comprehension task, that the models rely on the appearance of cue words in a language expression rather than its 'deeper' understanding. Jin et al. (2020) show the same but on text classification and natural language inference tasks.

The Problem of Efficiency in Neural Language Models:

Since the introduction of the word embeddings, the progress in NLP has been mainly driven by scaling both neural language models and training corpora to even larger sizes (Devlin et al., 2019; Liu et al., 2019b; Radford et al., 2019). As we discussed above, this approach was justified by reporting state-of-the-art performance on various tasks that require competence in understanding the meaning of language expressions. Furthermore, Warstadt et al. (2020) and Zhang et al. (2020b) show that there is a link between the amount of data Transformer-based (Vaswani et al., 2017) language models are trained on and their ability to acquire certain linguistic knowledge (e.g. subject-verb agreement). They found the more training data is provided the more linguistic knowledge it acquires. However, is this approach efficient at acquiring the competence in a natural language and whether it is the right way forward? There are some evidence that the answers to both questions are no.

Indeed, Warstadt et al. (2020) further assert that scaling amount of training data may not be the best way forward and we may need more efficient inductive biases in the model. Furthermore, Zhang et al. (2020b) observe that the models keep improving the performance on SuperGLUE tasks even after it has observed 30 billion of words and that it is likely to improve this performance with 100 times more data. However, if a model requires exponentially more data to become competent in the natural language understanding tasks then it becomes impractical to train such a model (Bender et al., 2021). Also, the enormous size of the models is not very justified. One argument is that the success of the distillation techniques (Sanh et al., 2019; Jiao et al., 2020) to a degree indicates that we may use smaller neural language models to achieve comparable performance to the larger models on the tasks.

Another argument that supports this is that simple scaling of the model size does not remedy some existing issues. For example, Lazaridou et al. (2021) state that it does not alleviate the issue of temporal generalisation.

A Way Forward:

The performance on the downstream tasks that we mentioned in the beginning and the linguistic knowledge that these models acquire is impressive and should not be diminished but more thorough evaluation techniques show that the current SOTA neural language models are far from human-level understanding of natural language expressions. Furthermore, the large size of the models and amount of data they require makes them inefficient. One may then ask how can we make models both efficient in terms of amount of data and model size and at the same time bridge the gap between humans and neural language models in terms of understanding of natural language expressions? In this thesis we hypothesis that a solution is incorporation of inductive biases. However, what inductive biases do we need? We focus on the following three inductive biases:[22] information-theoretic inductive bias, sparsity inductive bias, and a relational inductive bias encoded in knowledge graphs. In this section, we also talk about the structural inductive bias as we briefly discuss it in Chapter 5, though it is not the focus of the thesis. Below, we introduce each of the biases and discuss their importance in modelling of language expression.

2.3.5 Structural Inductive Bias

According to Chomsky (1965) to process a language expression one needs to prefer structural processing of the units of the expression over the linear (this is how the surface form of a language expression is presented to a learner). For example, a main verb of a sentence needs to agree with the subject of the sentence but not with its closest noun. To achieve this, a

[22] We need to acknowledge that there is the wide space of inductive biases and as it was discussed in McCoy et al. (2020a) there are numerous modelling decisions that can influence generalisations of a model. Hence, to make a study feasible one would need to make a choice of the biases he/she wishes to explore. We believe the ones that we choose to explore in this thesis are representative of this wide space and therefore help illuminate the advantages and disadvantages that inductive biases can bring to neural language modeling. In what follows we elaborate more on why we have chosen each of the biases.

neural network needs to relate the units (implicitly or explicitly) in a graph-like structure e.g. a tree graph. Since this theory has been influential in linguistics there have been attempts made to make neural models imitate this property by incorporating inductive biases. In this subsection, we discuss the most prominent approaches.

One approach to bias the neural models to prefer structural generalisation over the alternatives is via their architectural design. For example, RNNG (Dyer et al., 2016) and Tree-RNN[23] both allow explicit incorporation of a graph structure to guide the processing of the language expressions within the neural networks. Moreover, Shen et al. (2019) propose a new model, ON-LSTM, that extends the vanilla LSTM by introducing a new activation function and gating mechanism into the model that allows it to perform tree-like compositions. The benefit of ON-LSTM is that it does not require parse trees to be present in a dataset. However, the recent discussion brought by McCoy et al. (2020a) indicates that architectural inductive biases alone may not be enough to bias a model. As such RNNG and Tree-RNN may be a better alternative (Kuncoro et al., 2019; McCoy et al., 2020a). One disadvantage of these models is that it is hard to scale their training on a large corpus as it is quite expensive to annotate each sentence with a parse tree. Recently though, an unsupervised version of RNNG (Kim et al., 2019) was proposed that does not require the parse tree and it learns it by itself, but it is yet to match the performance of RNNG.

Bias injection via objective function is another approach. For example, Zhang and Hashimoto (2021) explain how the masked language model objective (Devlin et al., 2019), which is used to train SOTA neural language models, can bias the model to learn syntactic structures. The explanation is based on two arguments: First, they argue that there is a correspondence between the mask language model objective and Gaussian graphical model. They further elaborate that this association is why the model learns statistical dependencies between the units of a language expression. Second, they further argue that there is a close similarity between the statistical dependencies and syntactic dependencies, which explains

[23]Here, by Tree-RNN we mean a broad class of tree recurrent neural networks i.e. Tree-GRU (Chen et al., 2017), Tree-LSTM (Tai et al., 2015) including its specific instantiation of the model - Tree-RNN (Goller and Kuchler, 1996).

why the model can acquire (some) syntactic knowledge. Also, one can use a parsing objective to bias the structural generalisation (Dozat and Manning, 2017).

Finally, there are approaches that use data-based inductive biases. Such that, Min et al. (2020) introduce a data augmentation technique that allows to learn abstract syntactic representation for the models like BERT.

2.3.6 Sparsity Inductive Bias

In this subsection, we discuss how sparsity can allow us to model two properties of a natural language in neural networks: sparse "interaction" between language units and varying amount and type[24] of information contained in language units.

Let us elaborate more on the first property. Given a language unit such as sentence one can observe (performing either syntactic or semantic analysis of the sentence) that the interaction (both syntactic and semantic) between its smaller units - words - is sparse. For example, in English language main verb of a sentence needs to agree only with the subject and not with all the words in the sentence. The sparse interaction between language units can also be incorporated into a neural network in a form of inductive bias.

Modelling of the sparse interactions has especially been popular (Child et al., 2019; Correia et al., 2019; Ye et al., 2019; Zhao et al., 2019; Zhang et al., 2020a) in SOTA neural language model - Transformer. The vanilla Transformer models the interaction between the words/tokens via a fully connected graph (Joshi, 2020). However, this turns out to be not efficient both in terms of memory and time requirements.[25] Furthermore, it does not allow us to model sparse interaction between the units. To alleviate these issues, the most common approach has been sparsifying the Transformer by reducing a number of connections between the words/tokens. The former problem has been addressed by predefining the sparsity patterns in advance or limiting the way the units interact (Child et al., 2019; Ye et al., 2019). However, these approaches are not suitable to address the latter problem because depending on a sentence the interaction between units varies and cannot be predefined in advance. The

[24] By type we mean what information is conveyed by the unit, e.g. words may refer to different objects.
[25] Both of which grow quadratically with the length of a sequence.

solution to this problem was proposed by Correia et al. (2019), where the sparsity patterns adapt to a sentence.

Now lets discuss the second property in more detail. The amount and type of information conveyed by a language unit varies. As such, it makes sense to reflect this property in a vector representation of the sentence. Learning sparse representations of data can be dated back to Olshausen and Field (1996). This work motivates encoding of images in sparse linear codes for its biological plausibility and efficiency. Furthermore, it was later argued by Bengio (2009) that compared to the dimensionality reduction approaches, sparsity is a more efficient method for representation learning on vectors with fixed dimensionality for data of varying information content.

In NLP, learning sparse representations has been explored for various units of text with most of the focus placed on sparse representation of words. However, many of them only used sparsity to make the word embeddings and sentence embeddings (Trifonov et al., 2018) more interpretable (Faruqui and Dyer, 2015; Sun et al., 2016; Li and Hao, 2019), which is an important research direction but not of relevance for the current discussion. Although some of the works (Yogatama et al., 2015; Arora et al., 2018) employ sparsity to model properties of a natural language. Arora et al. (2018) use it to model polysemy and Yogatama et al. (2015) use sparsity to organise the dimensions of word embeddings into a hierarchical structure which in turn is a more biological plausible semantic representation (Collins and Quillian, 1969; Raposo et al., 2012). In Chapter 6 we further discuss how sparse representations may be a more natural way of modelling sentences in a fixed dimensional vector.

2.3.7 Information-Theoretic Inductive Bias

In various branches of linguistics: syntax, semantics morphology etc, information theory (Shannon, 1948) has been used to explain (or formulate) various language phenomena. In syntax, Head–Dependent Mutual Information hypothesis was put forward (Futrell et al., 2019) to explain the presence of a syntactic dependency between two words with the high mutual information between the words. In semantics, Zaslavsky et al. (2018) propose to use Information Bottleneck (IB; Tishby et al. (1999)) to explain how to efficiently assign surface

form of words to their meanings. In morphology, Cotterell et al. (2019) propose a new metric based on conditional entropy to quantify complexity of inflectional systems. In language production, its information-theoretic principle - Uniform Information Density (UID) - was proposed by Jaeger (2010). It states that information content should be distributed uniformly across a language expression.

Information theory, however, has not only been useful in explaining linguistic phenomena but also it has been used to design information-theoretic inductive biases for neural language models. For example, Wei et al. (2021) use UID as an inductive bias for better language modelling. The UID bias increases lexical diversity of generated text as well as improves language modelling perplexity. Mahabadi et al. (2021) use IB as a fine-tuning objective for low-resource language tasks. In these settings, they found IB to be useful regulariser that prevents overfitting and allows better generalisation to out-of-domain data. Furthermore, Wang et al. (2020) find IB to be a useful bias to improve the robustness of neural language models to adversarial attacks. In Chapter 5 we employ the Variational Autoencoder (VAE) framework to learn unsupervised sentence embeddings. We show how Kullaback-Leibler divergence can be used to regularise (or bias) the amount of information encoded in the sentence embeddings and demonstrate the effect that this bias has on the quality of the learned sentence representations using two downstream tasks: text generation and text classification.

2.3.8 Knowledge Graph Relational Inductive Bias

Knowledge Graphs (KGs) are a form of knowledge representation (Davis et al., 1993). They are shown to be useful as a standalone unit e.g., a KG can be used to predict new facts about the world (Nickel et al., 2015; Wang et al., 2017) it models as well as become key components in many NLP systems (Nastase, 2008; Pilehvar et al., 2013; Moro et al., 2014; Yih et al., 2015; Thorne et al., 2018). In this thesis, we view a KG as a form of an inductive bias that restricts the meaning of a language unit to the structure of the KG. That is, by defining the semantic relations between the units we strongly bias their meaning both by the choice of the relations and the choice of which units we connect together. This idea is similar to the one discussed in Battaglia et al. (2018), where various relational inductive biases are

discussed within a neural network model and how they bias the learning process. However, why we would need such an inductive bias?

Weissenborn (2017) argues that maintaining a large corpus that would contain all the relevant information about the world is not practical to train a neural network. Hence, there is no guarantee that all the required knowledge to solve a task would be presented in the training data. Moreover, Zhang et al. (2020b) remark on a large amount of data that is needed to learn common sense knowledge. However, if current SOTA models are not data efficient and keep all of the relevant information about the world in a corpus is not practical what would be an alternative? We argue that this alternative can be a KG - which can provide a strong inductive bias for a semantic representation. Since the semantic information in a KG is explicitly defined, the model may not need large amount of data to learn its meaning especially if the task is known.[26] Indeed it has been shown that the SOTA neural language models do indeed benefit from information incorporated in a KG (Zhang et al., 2019; Bauer et al., 2021). In Chapter 3 and 4 we show how to learn word and sentence embeddings respectively via KG.

[26]If the task is known we can use a KG design for that task

Part I

Data-Based Inductive Biases

3

Learning Word Embeddings with Relational Inductive Bias

3.1 Introduction[1]

Word embeddings can be seamlessly integrated into various NLP systems, effectively enhancing their generalisation power (Camacho-Collados and Pilehvar, 2018). However, as we discussed in Subsection 2.2.1, SOTA neural language models that are pretrained on large amount of text are unable to provide reliable representations for words such as domain-specific terms (Lazaridou et al., 2021) that are infrequent or unseen during training. One of the reasons for this is the lack of inductive biases that would allow the models to select the meaning of a word out of possible alternatives.

To address the unseen word representation problem, both for static and contextualised embeddings, several techniques have been proposed. Earlier works have mainly focused on morphologically complex words (Luong et al., 2013; Botha and Blunsom, 2014; Soricut and Och, 2015), whereas more recently, character-based and subword[2] unit information has garnered a lot of attention (Bojanowski et al., 2017) because of its ability to generalise to new words. In this case, out of the possible meanings that the unseen words can have, the semantics of the words is biased in terms of the units (characters, subwords, etc) of

[1]This chapter draws from the following publication: **Victor Prokhorov**, Mohammad Taher Pilehvar, Dimitri Kartsaklis, Pietro Lio, Nigel Collier (AAAI, 2019) "Unseen Word Representation by Aligning Heterogeneous Lexical Semantic Spaces".

[2]Note, that the algorithms that induce the subwords do not always capture what linguists would think of as morphemes.

the frequent words. Despite their success, such subword models make two assumptions around the unseen word: (1) variations of the word exist in the training corpus (for instance, occurrences of *track*—or even *untrack*—should exist to induce embeddings for *untracked*); and (2) the semantics of the word can be estimated based on its subword units which might not hold for single-morpheme words, e.g., *galaxy*, or for exocentric compounds (non-compositional compounds; also certain types of idiomatic expressions), e.g., *honeymoon*. As a result, they fall short of effectively representing the semantics of unseen single-morpheme words for which no variation has been observed during training, essentially ignoring most of the rare domain-specific entities which are crucial for NLP systems when applied to those domains (Pilehvar and Collier, 2016).

Alternatively, to learn a reliable semantic representation of a word, in the absence of large amounts of text containing this word, one can use a lexical resource i.e. dictionaries, KG, etc that encode the lexical knowledge of the word. The lexical resource gives an explicit definition of the words either in a form of text or in form of structural relations (e.g. hypernymy-hyponymy relations) that exist between the words. Thus can strongly bias the meaning of the words, which we can employ to guide the model to select the meaning of a word out of possible alternatives. There exist many high coverage and domain-specific[3] KG which contain valuable information for infrequent words. Recently, various embedding induction techniques have attempted to leverage lexical resources, such as WordNet (Bahdanau et al., 2017; Pilehvar and Collier, 2017) or Wikipedia (Lazaridou et al., 2017). Despite their success, they either rely on word definitions (glosses) or related words extracted from the lexical resource while ignoring the knowledge encoded in the semantic structure.

In this chapter, we present a methodology that exploits the semantic structure of the KG as a from of inductive bias[4] for unseen word representation. The technique first embeds a knowledge graph into a vector space and then maps the embedded words from this space to a corpus-based space, in order to expand the vocabulary of the latter with additional

[3]We should clarify that some domains such as biological/medical have hundred or so high quality high coverage KG, but others for example for engineering an aircraft or for understanding moon rocks may have none.

[4]More concretely, we bias the learning algorithm to derive a semantic representation of a word in terms of relationships that exist between the word (entity) and other words (entities) in a KG.

representations for rare and unseen words. We evaluate the reliability of our approach on several datasets across multiple tasks: six datasets for word similarity measurement and eight sentiment analysis and topic categorization datasets. Experimental results show that, unless ample occurrences exist in the training data, we can compute more reliable embeddings than the ones generated by state-of-the-art corpus based embedding techniques.

3.2 Background

Given its importance, unseen word representation has attracted considerable research attention for the past few years. Earlier techniques have mainly focused on improving distributional models for better handling of infrequent words (Sergienya and Schütze, 2015), or on inducing embeddings for morphological variations (Alexandrescu and Kirchhoff, 2006; Lazaridou et al., 2013; Luong et al., 2013; Botha and Blunsom, 2014; Soricut and Och, 2015). The latter branch often utilizes a morphological segmenter, such as Morfessor (Creutz and Lagus, 2007), in order to break inflected words into their components and to compute representations by extending the semantics of an unseen word's morphological variations.

More recently, character-based models have garnered a lot of attention. In these models words are broken down into subword units and characters (Bojanowski et al., 2017), usually irrespective of their morphological structure. An unseen word's representation is induced by combining the information for its subword units; for instance, by averaging the vector representations of its constituent character n-grams as done by FastText (Bojanowski et al., 2017). Character-based models have been successfully tested in different NLP tasks, including language modeling (Sutskever et al., 2011; Graves, 2013), part-of-speech tagging (Dos Santos and Zadrozny, 2014; Ling et al., 2015) and syntactic parsing (Ballesteros et al., 2015). However, all these techniques fall short of inducing representations for single-morpheme words that are not seen frequently during training as they base their modeling on information available from sub-word units. In contrast, our alignment-based model can also induce embeddings for single-morpheme words that are infrequent or unseen in the training data, such as domain-specific entities.

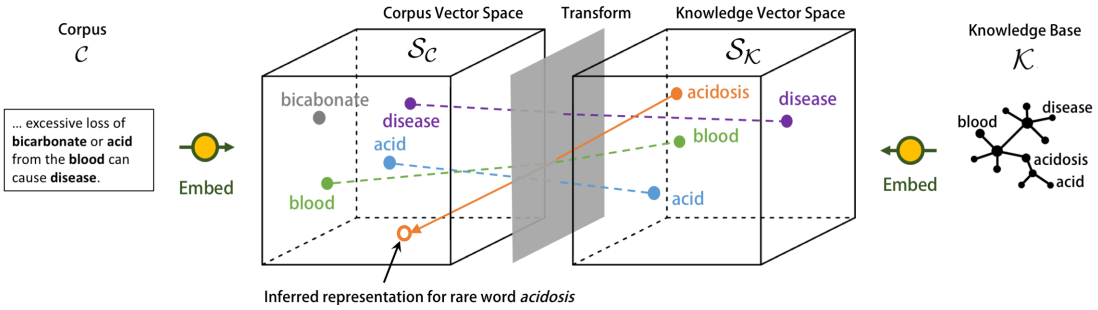


Fig. 3.1 Our coverage enhancement procedure. The dashed lines represent semantic bridges and the solid line represents a rare word that is projected from the knowledge vector space to the corpus vector space.

Most related to our work are the WordNet-based approaches of Pilehvar and Collier (2017) and Bahdanau et al. (2017). The former computes an unseen word's embedding by extracting the set of its semantically similar words ("semantic landmarks") from WordNet and combining their embeddings, whereas the latter trains a recurrent neural network, specifically, an LSTM, to estimate a word's embedding given its definition from WordNet. Moreover, the additive model of Lazaridou et al. (2017) is analoguous to the LSTM model (though less complex) and computes an embedding as the centroid of the embedding of the words in its definition. Despite addressing the single-morpheme word representation limitation of morphological models, these approaches ignore the information encoded in WordNet's lexical-semantic relations. We improve over these by proposing a model that effectively leverages the semantic network of WordNet.

3.3 Methodology

Figure 3.1 illustrates our procedure for enriching an existing corpus vector space S_C based on the lexical knowledge in an external knowledge graph K. The proposed algorithm mainly relies on techniques from two research areas: graph embedding and vector space transformation. Two main steps are involved in the process. First, it views K as a knowledge graph and transforms it to a vector space representation (S_K) by leveraging graph embedding techniques (Subsection 3.3.2). Then, it aligns the two vector spaces, i.e., KG- (S_K) and

corpus-based (S_C), by using vector space transformation algorithms (Subsection 3.3.3). As a result of this alignment, new embeddings are induced for unseen words in S_C. In our toy example in Figure 3.1, the term *acidosis* is missing from the vocabulary of S_C but it is covered by the knowledge graph K. First, a graph embedding algorithm is used to embed K, represented as a graph, into a vector space S_K. Then, based on common clues from the two spaces, a transformation function is learnt in order to map the vectors across the two spaces. The transformation function (from the embedded KG space S_K to the corpus space S_C) allows us to project the vector for *acidosis* to the latter space, hence inducing a new representation for the word.

3.3.1 Prerequisites

In our experiments, we used WordNet 3.0 (Fellbaum, 1998) as external knowledge graph. The resource contains around 120K groups of synonyms, referred to as *synsets*, which are connected to each other by means of around 200K lexical semantic relations, such as hypernymy and meronymy. We further enrich the network by connecting a synset to all other synsets that appear in its disambiguated gloss[5]. This approach more than doubles the number of edges in WordNet's semantic network. As for the corpus vector space, any distributional semantic representation can be used. In our experiments, we opted mainly for word embeddings (rather than conventional count-based representations) due to their popularity.

Our procedure requires two additional conditions. Let V_K and V_C be the respective vocabularies of knowledge graph and corpus vector spaces. The first condition to be met is that V_K and V_C should have overlapping words, i.e., $V_K \cap V_C \neq \phi$. This is required for enabling the alignment of the two spaces (to be discussed in Subsection 3.3.3). The second condition is that the knowledge graph K has to provide lexical knowledge for unseen or infrequent words in the corpus vector space. Thanks to the abundance of knowledge graphs and the long tail of words in distributional representations, this condition is not difficult to be fulfilled.

[5] wordnet.princeton.edu/glosstag.shtml

3.3.2 Knowledge Graph Embedding

The proposed coverage enhancement procedure starts by transforming the lexical knowledge representation in the knowledge graph K to a form which is comparable to the corpus-based representation S_C. To this end, we *embed* the structural lexico-semantic knowledge of K into a vector space S_K.

We opted for node2vec[6] (Grover and Leskovec, 2016), a random walk based graph embedding technique which has proven its potential in the reliable representation of graph nodes. Given a graph G, the algorithm first generates a stream of artificial "sentences" by performing a series of random walks over G. Each such "sentence" contains a sequence of "words" (i.e, vertices) such that consecutive words correspond to neighbouring vertices in G. Analogously to the natural language text in which semantically similar words are expected to appear in similar contexts, an artificial sentence encodes local information for a node from the graph by placing topologically close vertices in similar contexts. Representations are then computed for individual vertices by taking a similar objective to the Skip-gram model (Mikolov et al., 2013), i.e., by maximizing $\prod_{j=i-z, j\neq i}^{i+z} \mathbf{Pr}(w_j|w_i)$ which is the probability of a word w_i given its context, where z is the window size or the length of the random walk. The only difference from the original Skip-gram model lies in the way input "sentences" are constructed.

In our experiments, we set the parameters of node2vec as follows: walk length to 100, window size to 10, and embedding dimensionality to 100. To decide on these parameters, we carried out experiments on the MTURK-771 dataset (Halawi et al., 2012). Also, note that nodes in the semantic graph of WordNet represent synsets. Hence, a polysemous word would correspond to multiple nodes. In our word similarity experiments (Subsections 3.4.1 and 3.4.2) we use the *MaxSim* assumption of Reisinger and Mooney (2010) in order to map words to synsets: the similarity of two words is computed as that of their closest associated meanings. In the downstream experiment (Subsection 3.4.3), we compute a single word vector as the average of its corresponding synsets' vectors.

[6] https://github.com/snap-stanford/snap/tree/master/examples/node2vec

3.3.3 Vector Space Alignment

Once the KG K is represented as a vector space S_K, we project it to S_C in order to improve the word coverage of this space with additional words from S_K. In this procedure we make two assumptions. Firstly, the two spaces provide reliable models of word semantics; hence, the relative within-space distances between words in the two spaces are comparable. Secondly, there exists a set of shared words between the two spaces (also mentioned in Subsection 3.3.1); we refer to these words as *semantic bridges*.

For this transformation we opted for Canonical Correlation Analysis (Faruqui and Dyer, 2014; Upadhyay et al., 2016, CCA), which is widely used for the projection of spaces belonging to different languages with the purpose of learning multilingual semantic spaces. The model receives as input two vector spaces for two different languages and a seed lexicon for that language pair, and learns a linear mapping between the two spaces. Ideally, words that are semantically similar across the two languages will be placed in close proximity to each other in the projected space.

Specifically, let $S'_C \subset S_C$ and $S'_K \subset S_K$ be the corresponding subsets of semantic bridges, i.e., words that are monosemous according to the WordNet sense inventory, for corpus and KG spaces, respectively. Note that S'_C and S'_K form matrices that contain representations for the same set of words, i.e., $|S'_C| = |S'_K|$. CCA finds a linear combination of dimensions in S_C and S_K which have maximum correlation with each other. Given two column vectors S'_C and S'_K of embeddings in the two spaces, CCA computes vectors w_C and w_K such that the random variables $w_C S'_C$ and $w_K S'_K$ maximize the correlation $\rho(w_C S'_C, w_K S'_K)$:

$$
\begin{aligned}
w_C^*, w_K^* &= \mathbf{CCA}(S'_K, S'_C) \\
&= \arg\max_{w_C, w_K} \rho(w_C S'_C, w_K S'_K) \\
&= \arg\max_{w_C, w_K} \frac{w_C \Sigma_{CK} w_K}{\sqrt{w_C \Sigma_C w_C^T} \sqrt{w_K \Sigma_K w_K^T}}
\end{aligned}
$$

where Σ_X and $\Sigma_{X,Y}$ denote covariance and cross-covariance, respectively. Note that the maximization is invariant to scaling of w_C and w_K. Hence, we can have a constraint for unit variance:

$$w_C^*, w_K^* = \underset{w_C \Sigma_C w_C^T = w_K \Sigma_K w_K^T = 1}{\arg\max} w_C \Sigma_{CK} w_K$$

The dimensionality of the resultant space in our experiments is $\min(d_C, d_K) = d_K = 100$, where d_C and d_K are the dimensionalities of the corpus and KG spaces, respectively. The enhanced space S^* is obtained as the union of $w_C S_C$ and $w_K S_K$. Note that this procedure is slightly different from the one illustrated in Figure 3.1. The enriched space is a third space which is independent from the two initial spaces S_K and S_C.

As for the seed lexicon (the set of semantic bridges S'_C and S'_K), we used the set of monosemous words in the WordNet's vocabulary which are deemed to have the most reliable semantic representations in the corpus vector space. Of the 155K words in WordNet's vocabulary, around 128K are monosemous, which provides us with a large set of semantic bridges to use for the alignment step. However, in our experiments we found that a small subset of 5K semantic bridges is enough for achieving reliable transformations.

Graph embedding and space alignment. For this work we experimented with node2vec. We note that there is a rich literature for graph embeddings (Cai et al., 2017). A series of algorithms first construct an adjacency matrix of the graph and obtain embeddings by directly factorising this matrix (Roweis and Saul, 2000; Cao et al., 2015), whereas others employ deep learning techniques, such as autoencoders (Wang et al., 2016). Relation embedding techniques such as TransE (Bordes et al., 2013) and HOLE (Nickel et al., 2016) are not suitable candidates for our purpose since their focus is rather on the embedding of edges (as opposed to nodes). As noted before, for the space alignment we experimented with CCA which is a linear model of projection.

3.4 Experiments

In this section[7] we provide three different sets of experiments that were carried out to evaluate the reliability of our rare word embedding induction technique (which we will refer to as **ALIGN**). First, we report results for in-vitro evaluations on the Stanford Rare Word similarity dataset (Subsection 3.4.1) and in a simulated rare word similarity setting (Subsection 3.4.2). We then verify the reliability of our induced embeddings in two downstream NLP tasks, sentiment analysis and topic categorization. This experiment is detailed in Subsection 3.4.3.

3.4.1 Rare Word Similarity

The Stanford Rare Word (RW) Similarity dataset (Luong et al., 2013) has been regarded as a standard benchmark for evaluating embedding induction techniques. The dataset comprises 2034 pairs of infrequent words, such as *ulcerate-change* and *nurturance-care*. In the first evaluation, we use this benchmark to compare our model against recent rare word representation techniques.

Experimental setup. We experimented with two sets of word2vec (Mikolov et al., 2013) embeddings trained on two different corpora: (1) W2V-GN, the Google News (vocab: 3M, dim: 300)[8], and (2) W2V-WP, the Wikipedia corpus (Shaoul and Westbury, 2010) (vocab: 2.4M, dim: 300). As for comparison systems, we benchmark our results against four other approaches: (1) **SemLand** (Pilehvar and Collier, 2017) which extracts for an unseen word the set of its semantically related words from WordNet and induces an embedding by combining their embeddings; (2) the **Additive** model of Lazaridou et al. (2017) which takes the unseen word's definition as semantic clue and induces an embedding by adding (averaging) the embeddings of content words in the defintion; (3) **LSTM**-based strategy of Bahdanau et al. (2017) which is a more complex version of the additive model that relies on an LSTM network which receives as its input the WordNet definition of the unseen word; and (4)

[7] It is worth noting that the experiments in this Chapter were conducted in year 2018-2019 and techniques employed as SOTA embedding techniques were from that time.

[8] code.google.com/archive/p/word2vec/

Embedding	W2V-GN		W2V-WP	
	r	ρ	r	ρ
W2V-GN	0.44	0.45	0.41	0.43
+ Additive	0.46	0.48	0.41	0.43
+ SemLand	0.48	0.51	0.39	0.40
+ LSTM	0.48	0.50	0.40	0.40
+ ALIGN	0.48	0.48	0.42	0.42

Table 3.1 Pearson (r) and Spearman (ρ) correlation for our approach (ALIGN) on the RW dataset with two pre-trained sets of word embeddings, before and after enhancement with various methods. FastText-WP (trained on the Wikipedia corpus): $r = 0.44$, $\rho = 0.44$ and node2vec (without any alignment and independent from corpus embeddings): $r = 0.16$, $\rho = 0.16$.

FastText[9] (Bojanowski et al., 2017) which computes a word embedding by combining the embeddings of its sub-word character n-grams (see Section 3.2 for more details).

Results. Table 3.1 shows correlation performance on the dataset for the two pre-trained word embeddings, in their initial form and when enhanced with additional induced word embeddings. Among the two initial embeddings, W2V-GN provides a lower coverage (173 out-of-vocabulary words vs. 88 for W2V-WP) despite its larger vocabulary (3M vs. 2.4M). All enhanced embeddings attain near full coverage (over 99%), thanks to the vocabulary expansion offered by WordNet. Our approach (ALIGN) produces competitive performance across the two settings and according to both Pearson and Spearman correlation metrics. The performance ($r = 0.16$, $\rho = 0.16$) of node2vec, when independently applied to this dataset, is notably lower than that of the initial corpus embeddings. However, it is interesting to note that these non-optimal embeddings can better the performance of corpus embeddings when combined with them, showing the complementarity of the two sources of information. Moreover, we hypothesise that the non-optimality of node2vec embeddings can also be attributed to the poor quality of the dataset (Pilehvar et al., 2018). We evaluate the quality of

[9] Another alternative to FastText is to use a more powerful model - Transformers with Byte Pair Encoding (BPE) (Vaswani et al., 2017), however as the recent findings show it also perform poorly on induction of embeddings for rare words (Schick and Schütze, 2020; Lazaridou et al., 2021). Moreover, it has been shown (Yu et al., 2021) that the Transformer based model BERT (Devlin et al., 2019) benefits form dictionaries in task of induction embeddings for rare words.

node2vec embeddings on the other word similarity dataset and find that they are competitive with the two SOTA word embeddings: W2V-GN and GLOVE, see Appendix A.1.

Comparison with FastText. FastText proves competitive on the dataset ($r = 0.44$, $\rho = 0.44$), highlighting the effectiveness of induced word embeddings from sub-word (character) information. This is not a surprise given that around a third of the rare words in the RW dataset are plural or *-ed* forms which can be easily handled by resorting to the embedding of their singular or uninflected forms. For instance, *kindergarteners* and *postponements* are highly similar to their singular forms and the semantics of *encrusted* and *entrapped* can be estimated to a good extent from *encrust* and *entrap* which are relatively more frequent terms. None of the other models in the table have access to this information. However, as mentioned earlier, the sub-word backoff strategy might not be effective for single-morpheme words and exocentric compounds, which in a real-world scenario account for the most frequent cases of unseen words and can be effectively handled by our model.

Reliability of the RW dataset. The Stanford Rare Word Similarity dataset has been regarded as a standard evaluation benchmark for rare word representation and similarity, and as such it is included in the experiments of this chapter. However, the variance across the scores provided by different annotators for the same pair is generally high in this dataset. This is mainly due to the reliance of the dataset on crowdsourcing without having rigorous checkpoint on the raters. As also highlighted by Pilehvar et al. (2018), the low-confidence annotations are also reflected by contradictory instances, such as the two (almost) identical pairs *tricolour-flag* and *tricolor-flag* which have received the two very different scores of 5.80 and 0.71. Hence, further improvements on the dataset (over the W2V-GN baseline), provided by different techniques, cannot be meaningfully interpreted. Given the unreliability of the benchmark, in the following section, we provide an alternative evaluation based on standard (common) word similarity benchmarks.

Embedding	Setting	RG-65		SimLex-999		MEN-3000		SimVerb-3500		WS-353 Sim	
		r	ρ	r	ρ	r	ρ	r	ρ	r	ρ
Initial word2vec	$T = 10$	0.40	0.42	0.15	0.12	0.46	0.45	0.07	0.08	0.53	0.54
	$T = 20$	0.54	0.56	0.22	0.21	0.53	0.52	0.12	0.11	0.63	0.62
	$T = 50$	0.63	0.63	0.26	0.24	0.63	0.62	0.15	0.15	0.68	0.69
	$T = 100$	0.68	0.69	0.30	0.28	0.65	0.64	0.19	0.18	0.73	0.73
ALIGN	$T = 0$	**0.86**	**0.88**	**0.40**	**0.37**	0.65	0.66	**0.42**	**0.39**	0.71	0.69
LSTM	$T = 0$	0.52	0.57	0.19	0.19	0.19	0.20	0.28	0.29	0.18	0.21
Additive	$T = 0$	0.56	0.59	0.17	0.13	0.24	0.23	0.21	0.20	0.31	0.32
SemLand	$T = 0$	0.52	0.53	0.22	0.20	0.38	0.38	0.23	0.22	0.43	0.40
FastText	$T = 0$	0.77	0.80	0.32	0.32	**0.76**	**0.76**	0.22	0.21	**0.74**	**0.73**

Table 3.2 Results of corpus-based and enhanced embeddings in the simulated rare word similarity setting.

3.4.2 Simulated Rare Word Similarity

For a word similarity dataset to be suitable for this evaluation, it has to contain words that are infrequent in generic texts. However, most of the existing standard word similarity datasets contain only high frequency words, which makes them unsuitable for evaluating rare word representation techniques. To work around this limitation, we follow Sergienya and Schütze (2015) and leverage corpus downsampling in order to artificially transform standard word similarity datasets to rare word similarity benchmarks. This enables us to evaluate our embedding induction technique on a variety of standard datasets.

Experimental setup. Let T be the *rarity threshold*, i.e., the expected occurrence frequency of an artificial rare word in the training text corpus. We process the original text corpus in order to guarantee that each word in the similarity dataset appears at most T times in the training corpus. This can be achieved by replacing all but T occurrences of the word with another unique token (e.g., the word concatenated by some unique character). As a result of this procedure, we obtain a corpus for each T value and for each dataset. Training word embeddings on these corpora simulates a setting in which all the words in the word similarity dataset are rare as they occur infrequently in the training corpus. Except from the corpus downsampling step, the experimental setup is similar to that of the previous experiment.

Datasets. For this experiment, we opted for five standard word similarity datasets: RG-65 (Rubenstein and Goodenough, 1965), SimLex-999 (Hill et al., 2015b), MEN (Bruni et al., 2014), WordSim-353 similarity subset (Agirre et al., 2009), and SimVerb-3500 (Gerz et al., 2016) which contains verbs only.

Results. Table 3.2 lists correlation performance results on the five datasets and for four different values of T (10, 20, 50, and 100) for the initial downsampled W2V-WP embeddings[10] as well as for enhanced embeddings using different techniques for $T = 0$ (unseen word setting). As expected, there is a steady improvement for the corpus-based embeddings with increasing values of T. On all the datasets and according to both evaluation measures, ALIGN significantly improves over the three other WordNet-based approaches. Interestingly, our induced embeddings consistently outperform corpus embeddings which are constructed with $T = 10$, 20, and 50 on all the datasets and are often better or on par with $T = 100$. This means that our approach can produce embeddings that are as reliable as those corpus embeddings that are computed based on 100 occurrences. This is important as around 80% of the words in the vocabulary of the Wikipedia corpus appear fewer than 50 times in the whole corpus.[11] Moreover, surprisingly, on the SimVerb dataset the induced embeddings perform significantly better than the corpus-based embeddings, even at $T = 100$. This shows the superior quality of the induced verb embeddings, thanks to the hand-crafted part-of-speech-specific knowledge encoded for them in WordNet.

Similarly to the previous experiment, FastText proves to be a competitive baseline, outperforming our induced embeddings on two datasets. However, again, we note that FastText benefits from the advantage of having access to all plural forms of these (originally frequent) downsampled words in the training dataset, which might not establish a fair comparison. The simulated rare word similarity datasets address the unreliability issue of Stanford RW but still do not represent a real-world rare word scenario. Ideally, such a

[10] Obviously, for $T = 0$, word2vec would be unable to learn any embeddings, hence we do not show that setting.

[11] In the 2015 Wikipedia dump corpus with around 1.6B tokens, there are slightly over 1.9M word types with at least three occurrences. Of these word types, more than 80% appear at most 50 times in total, whereas more than two thirds of words in the vocabulary have frequency ≤ 20.

Initialization	Setting	Sentiment Analysis					Topic Categorization		
		PL04	PL05	RTC	IMDB	Stanford	BBC	NG	OH
X = 0%	Initial	**66.2**	75.4	**79.7**	85.4	80.4	**96.7**	86.5	27.8
	+ALIGN	63.7	**75.6**	79.4	**86.8**	**80.5**	96.5	**87.0**	**29.3**
X = 20%	Initial	**59.1**	67.2	63.8	71.1	70.1	93.1	67.4	16.4
	+ALIGN	58.9	**69.9**	**74.5**	**79.3**	**77.6**	**95.1**	**80.3**	**25.7**
X = 40%	Initial	**56.2**	63.5	62.7	70.3	66.1	91.0	62.8	15.7
	+ALIGN	55.6	**68.0**	**74.5**	**81.8**	**76.2**	**94.5**	**79.7**	**28.5**

Table 3.3 Accuracy performance on eight datasets for sentiment analysis and topic categorization. The best results for each setting are shown in bold. NG and OH stand for Newsgroups and Ohsumed, respectively.

dataset would contain named entities, domain-specific terms or other uncommon words that tend to appear infrequently in generic text corpora (which are often used for training word embeddings). We believe that rare word representation research requires such a high quality benchmark for more rigorous evaluations. We leave the possibility of the creation of such datasets to future work.

3.4.3 Evaluation in Downstream Tasks

We were also interested in having an *in-vivo* evaluation of the reliability of our induced embeddings in a real-world NLP system. Given that currently the most important application of word embeddings is in the initialization of the input layer in neural networks, we opted for a standard neural system as our evaluation benchmark.

Experimental setup. We experimented with a neural text classification system applied to two tasks: sentiment analysis (binary classification) and topic categorization (multi-class classification). The embedding layer of this system is initialized with pre-trained word2vec embeddings. Let L be the vocabulary of a given dataset. We dropped the pre-trained corpus embeddings for $X\%$ of the words in L and replaced them with our induced embeddings. We experimented with three X values: 0 (in which we used all the corpus embeddings to initialize the layer; new embeddings were induced to further improve coverage for those words missing in corpus embeddings' vocabulary), 20 and 40 (in which, respectively, 20%

and 40% of corpus embeddings were dropped, i.e., their corresponding words were treated as out of vocabulary). We were mainly interested in observing if the induced embeddings, first, could improve over corpus embeddings and, second, were able to re-gain system performance lost when dropping a part of the corpus embeddings. In all settings the embedding layer was not updated during training (static). This allows us to have a direct evaluation on the reliability of embeddings, independently from any updates and alteration they can undergo during training. In each configuration we repeat the experiment three times and report the average performance.

Text classification system. In our experiments, we used a CNN text classifier which is similar to that of Kim (2014). The only difference is that in our model, instead of directly inputting the pooled features from the convolutional layer to a fully connected softmax layer, they are first passed through a recurrent layer in order to enable a better capturing of long-distance dependencies. Specifically, as our recurrent layer we used LSTM (Hochreiter and Schmidhuber, 1997).

Datasets. For sentiment analysis we used five standard datasets, including PL04 (Pang and Lee, 2004), PL05 (Pang and Lee, 2005),[12] RTC[13], and IMDB (Maas et al., 2011) which are all binary datasets (with positive and negative labels) containing snippets of or full movie reviews. We also experimented with Stanford Sentiment dataset (Socher et al., 2013) which associates phrases with values that denotes their sentiments. To be consistent with the other four datasets' binary classification setting, we removed the neutral phrases with scores 0.4 to 0.6 and considered the reviews with values below 0.4 as negative and above 0.6 as positive. For the topic categorization task we used two newswire datasets: The BBC news dataset CR [14] (Greene and Cunningham, 2006) and Newsgroups (Lang, 1995) with 5 and 20

[12] Both PL04 and PL05 are obtained from http://www.cs.cornell.edu/people/pabo/movie-review-data/

[13] http://www.rottentomatoes.com

[14] http://mlg.ucd.ie/datasets/bbc.html

classes, respectively. We also experimented with a domain-specific categorization dataset: Ohsumed[15], which contains medical texts categorized into 23 classes.

Results. Table 3.3 shows the results. We report classification accuracy for the baseline system ("Initial") which is initialized by full ($X = 0\%$) or partial ($X > 0\%$) corpus-based embeddings, and for the enhanced systems with additional induced embeddings ("+ALIGN"). Generally, the enhancement proves to be beneficial as it provides improvements in most of the configurations across the eight datasets. In the $X = 0\%$ setting, the improvement is particularly noticeable for the IMDB, Newsgroup and Ohsumed datasets which have a fair portion of their vocabularies not covered by word2vec embeddings. However, lower or no improvement is observed for other datasets (particularly, PL04) whose vocabularies are largely covered by the corpus embeddings. In the $X > 0\%$ settings, the performance of the baseline system drops significantly on most datasets. In the 20% setting, which is the closest to a real-world scenario, the enhanced system can recover a large part of the lost performance on most of the datasets. The same trend is observed for $X = 40\%$. Interestingly, on the Ohsumed dataset, which belongs to the medical domain, the enhanced system gets close to the initial system initialized by corpus embeddings. This is a strong indication of the effectiveness of our approach in filling lexical gaps for specific domains. Overall, the results show that our induced embeddings, though not sufficient to replace corpus embeddings for frequent words, can significantly improve over infrequent or unkown embeddings, particularly for specific domains.

3.5 Conclusions and Future Work

To learn reliable word embeddings SOTA models need large amount of text (frequently) containing these words. One of the reasons for this is the lack of inductive biases that would allow the models to select the meaning of a word out of possible alternatives. Hence, words that are not frequent or absent in the text cannot be represented reliably with the embedding.

[15] ftp://medir.ohsu.edu/pub/ohsumed

However, can we use an inductive bias that allows us to use much smaller amount of data and still learn a good representation for these rare and unseen words?

In this chapter, we investigated if a KG can be used as such inductive bias. Such that, we presented a methodology for marrying distributional semantic spaces with lexical knowledge graphs and applied it to the task of extending the vocabulary of the former with the help of information extracted from the latter. By evaluating the induced embeddings on multiple word similarity benchmarks as well as on a downstream NLP evaluation framework, we showed that the KG is a reliable inductive bias to learn the semantic representation of words.

In future work, we plan to experiment with domain-specific KGs, such as medical KGs, and study the efficacy of our methodology. Moreover, we plan to further experiment with non-linear transformation techniques[16], such as Kernel CCA (Akaho, 2006) and Deep CCA (Andrew et al., 2013) and also explore how can we learn representations of words that are in KG and words that are in a corpus, jointly.

In the next chapter, we extend the idea of biasing semantic representation of words to phrases/sentences, where we map a dictionary definition to path graphs extracted from a KG.

[16]Most of the cross-lingual (our work can be thought as a type of cross-lingual mapping) word embedding mappings assume that the two vector spaces are isomorphic (or assume that their structure is similar) hence use linear transformation, however if two vector spaces are not similar then the linear transformation has its limitations (Ormazabal et al., 2019). In this work, this assumption is also reasonable as we discuss in Subsection 3.3.3. However, if one uses KG and text from the different domains e.g. a KG is in engineering and text is a news corpus then this assumption may no longer be valid. Hence, non-linear mapping may perform better (Glavaš and Vulić, 2020). One potential difficulty with using powerful linear mappings proposed above is availability of large quantity of data. Thus one may need to find a KG and text where there will be plenty of semantic bridges between the two vector spaces.

4

Learning Sentence Embeddigns with Relational Inductive Bias

4.1 Introduction[1]

In the previous chapter, we discussed how a KG can be used as an inductive bias to learn word embeddings. Inspired by the work of Hill et al. (2015a) and Hill et al. (2016), in this chapter, we present a work that demonstrates how one can further use relations between entities in a KG as a relational inductive bias for learning phrase/sentence[2] embeddings.

Learning semantic representation of sentences is an immensely hard task because numerous possible meanings can be expressed by composing the words in the sentences. One way to alleviate this issue is via a supervision signal that expresses (or biases) the meaning of the sentences. However, what would be the 'right' supervision signal to learn the meaning of sentences? Hill et al. (2015a) propose to learn embeddings of phrases by mapping dictionary definition to word embeddings. Here, we take this idea one step further and instead map unrestricted text to the sequence of entities in a KG. As a first step towards investigating whether this is a right bias to learning the meaning of sentences we pose the following question: can we bias semantic representation of a sentence to be reflective of topological dependencies that exist in Knowledge Graph (KG)? To perform the mapping we formulate this approach as the text-to-entity mapping.

[1] This chapter draws from the following publication: **Victor Prokhorov**, Mohammad Taher Pilehvar and Nigel Collier (NAACL, 2019) "Generating Knowledge Graph Paths from Textual Definitions using Sequence-to-Sequence Models".

[2] Textual definitions of concepts (or nodes of the graph) in a KG are defined via both sentences and phrases, however in this chapter, for ease of reading, we refer to both (phrase/sentence) as sentences.

Text-to-entity mapping is the task of associating a text with a concept in a knowledge graph (KG) or an ontology (we use two terms, interchangeably). Recent works (Hill et al., 2015a; Kartsaklis et al., 2018) use neural networks to project a text to a vector space where the entities of a KG are represented as continuous vectors. Despite being successful and also being able to bias semantic representation of a sentence, these models have two main disadvantages. First, they rely on a predefined vector space which is used as a gold standard representation for the entities in a KG. Therefore, the quality of these algorithms depends on how well the vector space is represented. Second, these algorithms are not interpretable; hence, it is impossible to understand why a certain text was linked to a particular entity which makes it hard to probe semantic information that is encoded in the sentence embedding.

To address these issues we propose a novel technique that first represents a KG concept as a sequence of its ancestors in the KG (hypernyms) and then maps the corresponding textual description to this unique representation. For example, given the textual description of the concept *swift* ("small bird that resembles a swallow and is noted for its rapid flight"), we map it to the hierarchical sequence of entities in a KG: *animal* $\rightarrow$ *chordate* $\rightarrow$ *vertebrate* $\rightarrow$ *bird* $\rightarrow$ *apodiform_bird*. This sequence of nodes constitutes a path.[3]

Our model is based on a sequence-to-sequence neural network (Sutskever et al., 2014) coupled with an attention mechanism (Bahdanau et al., 2014). Specifically, we use a LSTM (Hochreiter and Schmidhuber, 1997) encoder to project the textual description into a vector space and a LSTM decoder to predict the sequence of entities that are relevant to this definition. With this framework, we do not need to rely on the pre-existing vector space of the entities, since the decoder explicitly learns topological dependencies between the entities of the KG. Furthermore, the proposed model is more interpretable. Instead of the closest points in a vector space, it outputs paths; therefore, we can trace all predictions the model makes. In this chapter, we consider rooted tree graphs[4] only and leave the extension of the algorithm for more generic graphs to future work.

[3]We only consider hypernymy relations, from the root to the parent node (*apodiform_bird*) of the entity *swift*.

[4]Only a single root is allowed. If a tree has more than one root, one can create a dummy root node and connect the roots of the tree to it.

We evaluate the ability of our model in generating graph paths for previously unseen textual definitions on seven KGs (Section 4.3). We further demonstrate that our technique either outperforms or performs on a par with a competitive multi-sense LSTM model (Kartsaklis et al., 2018) by better utilising external information in the form of word embeddings. We use these results as the indicators that sentence embeddings do incorporate the semantics of the pathgraph; in this case the semantics is the hypernymy hierarchy of concepts.

4.2 Methodology

We assume that a KG is represented as a rooted tree graph $G = (V, E, T)$, where V is a set of entities (e.g. synsets in WordNet), E is a set of hyponymy edges, and T is a set of textual descriptions such that $\forall v \in V$ there is a $t_v \in T$.

4.2.1 Node representation

We assume that a KG concept can be defined by either using a textual description from a dictionary or hypernyms of the defining concept in the KG. For example, to define the noun *swift* one can use the dictionary definition mentioned previously. Alternatively, the concept of *swift* can be understood from its hypernyms, e.g. in the trivial case one can say that *swift* is an *animal*. This definition is not very useful since *animal* is a hypernym for many other nouns. To provide a more specific definition, one can use a sequence of hypernyms e.g. *animal* $\rightarrow$ *chordate* $\rightarrow$ *vertebrate* $\rightarrow$ *bird* $\rightarrow$ *apodiform_bird* starting from the most abstract node (root of a KG) to the most specif (parent node of the noun).

More formally, for each entity $v \neq v_{root} \in V$ we create a path p_v. Each p_v starts from v_{root} and ends with a hypernym of v, i.e., the hierarchical order of entities is preserved. Then the path p_v is aligned with t_v such that each node is defined by a textual definition and a path. This set of aligned representations is used to train the model.

The path representation of an entity ends with its parent node. Therefore, a leaf node will not be present in any of the paths. This is problematic if a novel definition should be attached to a leaf. To alleviate this issue we employ the “dummy source sentences" technique from

neural machine translation (NMT) (Sennrich et al., 2016a). We create an additional set of paths from the root node to each leaf. As for the textual definition we leave it empty.

4.2.2 Model

We use a sequence-to-sequence model with an attention mechanism to map a textual description of a node to its path representation.

Encoder. To encode a textual definition $t_v = (w_i)_{i=1}^{N}$, where N is sentence length, we first map each word w_i to a dense embedding e_{w_i} and then use a bi-directional LSTM to project the sequence into a latent representation. The final encoding state is obtained by concatenating the forward and backward hidden states of the bi-LSTM.

Decoder. Decoding[5] the path representation of a node from the latent state of the textual description is done again with an LSTM decoder. Similarly to the encoding stage, we map each symbol in the path $p_v = (s_j)_{j=1}^{M}$ to a dense embedding e_{s_j}, where M is the path length. To calculate the probability of the path symbol s_j at time step j we first represent the path sequence as $h_j^* = \text{LSTM}(e_s^j, h_{j-1}^*)$. Then, we concatenate h_j^* with the context vector c_j (defined next) and pass the concatenated representation $[h_j^*; c_j]$ through the softmax function, i.e. $s_j = \max(\text{softmax}(\mathbf{W}[h_j^*; c_j]))$, where $\mathbf{W}$ is a weight parameter. To calculate the context vector c_j we use an attention mechanism, $e_{ji} = v_a^T \tanh(\mathbf{W_a} h_i + \mathbf{U_a} h_j^*)$ and $c_j = \sum_i^N \text{softmax}(e_{ji}) h_i$, where v_a, $\mathbf{W_a}$ and $\mathbf{U_a}$ are the weight parameters, over the words in the text description.

[5]Note, potentially, our model can decode paths that do not exist in a KG. This is because, at each decoding step, the model outputs probability distribution over all nodes in KG. An alternative could be masking nodes that are not neighbours of the currently decoded node. This, potentially, should boost the performance of our model, however we leave testing of this hypothesis for future work. Presently, we test if a sequence-to-sequence model can be a competitive text-to-entity model without prior knowledge of a topology of a KG. See Subsection 4.3.4 for further discussion.

Graphs	|V|	Depth	Branch	A.D
PATO	1742	(4.94,10)	(3.95,92)	20
$WN_{animal.n.01}$	3999	(6.94,12)	(3.79,52)	26
$WN_{plant.n.02}$	4487	(4.70,9)	(5.91,357)	28
HDO	9095	(5.92,12)	(4.59,222)	27
HPO	13348	(6.95,14)	(3.40,32)	24
GO	29682	(6.40,14)	(3.28,172)	21
$WN_{entity.n.01}$	74374	(8.01,18)	(4.52,402)	36

Table 4.1 Statistics of the Graphs. $|\mathbf{V}|$ is the number of nodes, *depth* is the path length from the root of a graph to a node, *branch* is the number of neighbours a node has (leaves were removed from the calculation). The first value in the parentheses corresponds to the average and the second to the maximum value. A.D stands for average number of decisions the model makes to infer a path, i.e A.D = average depth $\times$ average branch.

4.3 Experimental Setup

KGs. We experimented with seven graphs four of which are related to the bio-medical domain: Phenotype And Trait Ontology[6] (PATO), Human Disease Ontology (Schriml et al., 2012, HDO), Human Phenotype Ontology (Robinson et al., 2008, HPO) and Gene Ontology[7] (Ashburner et al., 2000, GO). The other three graphs, i.e. $WN_{animal.n.01}$[8], $WN_{plant.n.02}$ and $WN_{entity.n.01}$ are subgraphs of the WordNet 3.0 (Fellbaum, 1998). We present the statistics of the graphs in Table 4.1.

KG Preprocessing. All the KGs we experimented with are represented as directed acyclic graphs (DAGs). This creates an ambiguity for node path definitions since there are multiple pathways from a root concept to other concepts. We have assumed that a single unambiguous pathway will reduce the complexity of the problem and leave the comparison with ambiguous pathways (which potentially would involve a more complex model) to future work. To convert a DAG to a tree we constrain each entity to have only one parent node. The edges between the other parent nodes are removed.[9]

[6] http://www.obofoundry.org

[7] After prerocessing GO we took its largest connected component.

[8] The subscript in 'WN' indicates the name of the root node of the graph.

[9] The choice of an edge is performed on random basis. An alternative would be to hire a domain expert who can determine which edges can be removed to reduce overall effect on the KG structure. However, this

Path Representations. We also experiment with two path representations. Our first approach, *text2nodes*, uses the label of an entity (Section 4.1) to represent a path. This is not efficient since the decoder of the model needs to select between all of the entities in a KG and also requires more parameters in the model. Our second approach, *text2edges*, to reduce the number of symbols for the model to choose from, uses edges to represent the path. To do this we create an artificial vocabulary of the size $\Delta(G)$, where $\Delta(G)$ corresponds to the maximum degree of a node. Each edge in the graph is labeled using the artificial vocabulary. For the example in Section 4.1, the path would be *animal* $-$[a]$\rightarrow$ *chordate* $-$[b]$\rightarrow$ *vertebrate* $-$[c]$\rightarrow$ *bird* $-$[d]$\rightarrow$ *apodiform_bird* where {a,b,c,d} is the artificial vocabulary. In the resulting path we discard labels for the entities; therefore, the path reduces to: [a]$\rightarrow$ [b]$\rightarrow$ [c]$\rightarrow$ [d].

4.3.1 Baselines

Bag-of-Words Linear Regression (BOW-LR). To represent a textual definition in a vector space we first use a pre-trained set of word embeddings (Speer et al., 2017) to represent words in the definition and then find the mean of the word embeddings. As for the KG, we use node2vec (Grover and Leskovec, 2016), to represent each entity in a vector space. To align the two vector spaces we use linear regression.

Multi-Sense LSTM (MS-LSTM). Kartsaklis et al. (2018) proposed a model that achieves state-of-the-art results on the text-to-entity mapping on the *Snomed CT*[10] dataset. The approach uses a novel multi-sense LSTM, augmented with an attention mechanism, to project the definition to the KG vector space. Additionally, for a better alignment between the two vector spaces, the authors augmented the KG graph with textual features.

approach would not be practical because of costs: time it would take for the expert to preprocess such a graph and money to hire such an expert. Even for the smallest graph (preprocessed) used in this work, on average, there are 1742*92=160,264 edges to consider. Also, in Appendix B we report the average number of nodes that have more than one parent and the average number of parents the nodes have (which is around 2 for all the KGs except PATO and GO, which is around 3).

[10] https://www.snomed.org/snomed-ct

4.3.2 Evaluation Metric

To perform evaluation of the models described above we used Ancestor-F1 score (Mao et al., 2018). This metric compares the ancestors ($is-a_{model}$) of the predicted node with the ancestors ($is-a_{gold}$) of the gold node in the taxonomy.

$$P = \frac{|is-a_{model} \wedge is-a_{gold}|}{|is-a_{model}|},$$

$$R = \frac{|is-a_{model} \wedge is-a_{gold}|}{|is-a_{gold}|},$$

where P and R are precision and recall, respectively. The Ancestor-F1 is then defined as:

$$2 \times \frac{P \times R}{P+R}.$$

4.3.3 Intrinsic Evaluation

To verify the reliability of our model on text-to-entity mapping we did a set of experiments on the seven graphs (Section 4.3) where we map a textual definition of a concept to a path. To conduct the experiments we randomly sampled 10% of leaves from the graph. From this sample, 90% are used to evaluate the model and 10% are used to tune the model. The remaining nodes in the graph are used for training. We sample leaves for two reasons: (1) to predict a leaf, the model needs to make the maximum number of (correct) predictions and (2) this way we do not change the original topology of the graph. Note that the sampled nodes and their textual definitions are not present in the training data.

Both baselines predict a single entity instead of a path. To have the same evaluation framework for all the models, for each node predicted by the baselines we create[11] a path from the root of the node to the predicted node. However, we want to emphasize that this is disadvantageous for our model, since all the symbols in the path are predicted by it and in the case of the baselines only a single node is predicted.

[11] We used NetworkX (https://networkx.github.io) to find a path from predicted node to the root of a graph.

Models	PATO	$WN_{animal.n.01}$	$WN_{plant.n.02}$	HDO	HPO	GO	$WN_{entity.n.01}$
BOW-LR	0.79	0.75	0.65	0.55	0.63	0.32	0.41
MS-LSTM$_{\lambda=0}$	0.77	0.73	0.62	0.70	0.72	0.69	0.51
MS-LSTM$_{\lambda=0.5}$	0.80	0.76	0.65	0.70	0.73	0.70	0.57
MS-LSTM$_{\lambda=1}$	0.75	0.66	0.57	0.65	0.63	0.62	0.51
text2nodes	0.75	0.66	0.66	0.69	0.62	0.67	0.60
text2edges	0.76	0.68	0.66	0.69	0.69	0.69	0.61
MS-LSTM$^{*}_{\lambda=0.5}$	0.81	0.76	0.66	0.71	0.74	0.71	0.58
text2nodes*	0.83	0.71	0.68	0.71	0.69	0.70	0.62
text2edges*	**0.83**	**0.77**	**0.70**	**0.73**	**0.74**	**0.72**	**0.65**

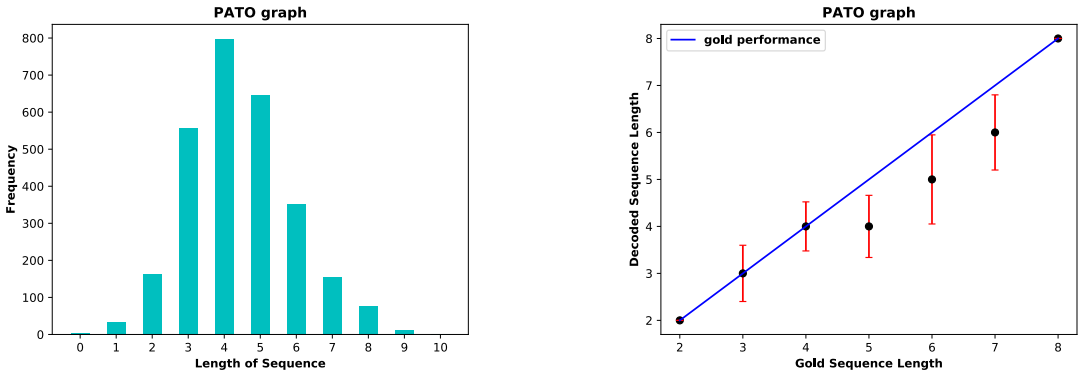
Table 4.2 Ancestor F1 results. Numbers in bold represent the best performing system on a graph. Models marked with $*$ make use of pre-trained word embedding in their encoder. Lambda (λ) is defined in Subsection 4.3.1. We use the same number of epochs, batch size and number of latent dimensions both for MS-LSTM and our models (Appendix B.2).

The results are presented in Table 4.2. Models that are in the last three rows of Table 4.2 use pre-trained word embeddings (Speer et al., 2017) in the encoder. MS-LSTM and our models that are above the last three rows use randomly initialised word vectors. We had four observations: (1) without pre-trained word embeddings in the encoder our model outperforms the best MS-LSTM$_{\lambda=0.5}$ only on two of the seven graphs, (2) the text2edges* model outperforms all the other models including MS-LSTM$^{*}_{\lambda=0.5}$, (3) the text2edges model can better exploit pre-trained word embeddings than MS-LSTM, (4) our model performs better when the paths are represented using edges (rather than nodes). We also found that there is a strong negative correlation (Spearman: -0.75, Pearson: -0.80) between A.D. (Table 4.3) and the Ancestor F1 score for the text2edges* model, meaning that with an increase in A.D. the Ancestor F1 score decreases.

4.3.4 Error Analysis

We carried out an analysis on the outputs of our best-performing model, i.e. text2edges* with pre-trained word embeddings. One factor that affects the performance is the number of invalid sequences predicted by the text2nodes and text2edges models. An invalid sequence is the path that does not exist in the original graph. This happens because at each time step the decoder outputs a distribution over all the nodes/edges and not just over possible children nodes. We

therefore performed a count of the number of invalid sequences produced by the model. The percentage of invalid sequences is in the range of 1.82% - 8.50% (Appendix B.1.1), which is relatively low. This analysis was also performed by J. Kusner et al. (2017). To guarantee that the model always produces valid graphs, they use a context-free grammar. A similar method can be adapted in our work.



(a) (b)

Fig. 4.1 The graph on top shows the length of sequence vs length frequency on a training set. The graph on the bottom shows the length of the gold sequence vs mean length of decoded sequence on the test set.

Another factor that affects the performance is the length of the generated paths which is expected to match the length of the gold path. To test this, we compared the mean length of the generated sequences with the length of the gold path (the graph on the bottom of Figure 4.1). Also, in the training set, we associate the length of the sequences with their frequencies (the graph on the top of Figure 4.1). We found that (1) the length of the generated paths are biased towards the more frequent paths in the training data, (2) if the length of a path is not frequent in the training data, the model either under-generates or over-generates the length (Appendix B.3).

4.4 Conclusion and Future Work

One of the existing problems in the learning of phrase/sentence embeddings is to find a task that would allow us to assign a meaning to the composition of the words. It is a complex

task because numerous possible meanings can be expressed by composing the words in the sentences/phrases. A potential task would include a supervision signal that would explicitly express a meaning of the phrases and sentences. However, what can we use as a supervision signal to express the meaning?

We followed the work of Hill et al. (2015a) who proposed to relate the meaning of phrases/sentences with words via a lexical resource. We presented a model that biases the semantic representation of sentences in terms of relationships that exist between the entities in a KG and used intrinsic experiments to demonstrate this. We evaluated the proposed technique on seven KGs: 1) showing that it can bias semantic representation of a sentence to be reflective of topological dependencies that exist in the KGs, and 2) it performs competitively with respect to existing SOTA text-to-entity systems, while being more interpretable and self-contained.

We have indirect evidence that this inductive bias may lead to better generalisation than alternative techniques that learn sentence embeddings (Hill et al., 2016). However, we leave it for future investigation. One reason for this is that modern NLP is driven by a large amount of data and models, which in turn require a lot of computing power and resources (Liu et al., 2019b). As such to make this further experiment meaningful (to adequately compare with SOTA models) we will require to train (or fine-tune) these models with large KG which is beyond our compute resource.

A natural next step will be to extend our framework to DAGs and use a pretrained Transformer based neural language model instead of LSTMs, and also testing the generalisation ability of the models on downstream tasks. We also hope that this work will motivate further exploration of KG as a data-based inductive bias for learning sentence embeddings.

In Chapter 3 and this chapter, we used data (data-based inductive biases) to bias representations of words and sentences. Despite being effective, creation of the data i.e. corpora, KG, etc with labels (or certain properties as in Andreas (2020)) that would allow us to bias the embeddings of words/sentences can be and expensive and time-consuming process. In the next two chapters, instead, we incorporate an inductive bias into the model itself (data-agnostic inductive biases).

Part II

Data-Agnostic Inductive Biases

5

Learning Sentence Embeddings with Information-Theoretic Inductive Bias

5.1 Introduction[1]

Due to the flexibility of (un/self)supervised representation learning the use of autoencoder neural architectures received a lot of attention. In principle, autoencoders try to preserve as much as possible information about the data they model (Valpola, 2014). However, as an inductive bias it is poorly understood what kinds of implications the amount of information would have on downstream tasks. In this chapter, we explore this question for learning representation of sentences using Variational Autoencoder (VAE) framework.[2]

The vanilla VAE (Kingma and Welling, 2014) applied to text has been shown to be a promising framework for learning sentence embeddings (Bowman et al., 2015b). It consists of an encoder (inference or approximate posterior) and decoder (generative) networks: Given an input x, the encoder network parameterizes $q_\phi(z|x)$ and infers about latent continuous representations of x, while the decoder network parameterizes $p_\theta(x|z)$ and generates x from the continuous code z. The two models are jointly trained by maximizing the Evidence

[1]This chapter draws from the following publication: **Victor Prokhorov**, Ehsan Shareghi, Yingzhen Li, Mohammad Taher Pilehvar and Nigel Collier (WNGT, 2019) "On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation".

[2]Here we take an information-theoretic view of VAE and link it to mutual information. Potentially, similar questions that we investigate here can be studied using other formulation of mutual information maximisation principle (Barber and Agakov, 2003; Hjelm et al., 2019; Kong et al., 2020). However, with these frameworks discriminative (Li et al., 2021) and generative (Zhang et al., 2018; Pan et al., 2020) tasks, to best of our knowledge, are studied separately, but VAE allow us to study both these tasks using the same model.

Lower Bound (ELBO), $\mathscr{L}(\theta, \phi; x, z)$:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\big(q_\phi(z|x)||p(z)\big) \tag{5.1}$$

where the first term is the reconstruction term, and the second term is the Kullback-Leibler (KL) divergence between the posterior distribution of latent variable z[3] and its prior $p(z)$ (i.e., $N(0, I)$). The KL term can be interpreted as a regularizer which prevents the inference network from copying x into z, and for the case of a Gaussian prior and posterior has a closed-form solution.

In this chapter, we propose to use an extension of VAE (Burgess et al., 2018) which permits us to explicitly control the magnitude of the KL term. We show that by controlling this term we can bias the amount of information that is encoded in the sentence embedding. We use an existing theoretical framework (see Section 5.2) as well as empirical results that support this claim (see Subsection 5.3.1). Since we can control the amount of information a VAE model encodes in the sentence embeddings, during the learning, it can be treated as an inductive bias. We study the implications this information-theoretic inductive bias has on components (generative and inference networks) of the VAE model via intrinsic analysis of the components and performance of the model on downstream tasks.

First, we study how the amount of information that is encoded in the sentence embeddings affects the shape of the approximate posterior as well as the proximity of aggregated posterior (see Subsection 5.3.2) to the prior distribution. Then we conduct a set of qualitative and quantitative experiments analysing the effect this inductive bias has on the generative capacity of VAEs. Moreover, we establish a link between the discriminative performance of latent sentence representations (on three text classification tasks) and the amount of information that is encoded in the representations. Finally, we test if biasing the amount of information in the sentence embeddings results in the encoding of structural[4] signal (see Subsection 2.3.5) in them.

[3] Here, the latent variable represents the sentence embedding. We use the two interchangeably.

[4] We test for the presence of syntactic information. This also opens a broader discussion on what information, about a sentence, should be modelled globally by the encoder and what information should be modelled locally by the decoder (Chen et al., 2016).

5.2 Information-Theoretic View of VAE

We take the encoder-decoder of VAEs as the sender-receiver in a communication network. Given an input message x, a sender generates a compressed encoding of x denoted by z, while the receiver aims to fully decode z back into x. The quality of this communication can be explained in terms of *rate* (R) which measures the compression level of z as compared to the original message x, and *distortion* (D) which quantifies the overall performance of the communication in encoding a message at the sender and successfully decoding it at the receiver. Additionally, the capacity of the encoder channel can be measured in terms of the amount of mutual information between x and z, denoted by $\mathrm{I}(x;z)$ (Cover and Thomas, 2012).

5.2.1 Reconstruction, KL and Mutual Information

The reconstruction loss can naturally measure distortion ($D := -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$), while the KL term quantifies the amount of compression (rate; $R := D_{KL}[q_\phi(z|x)||p(z)]$) by measuring the divergence between a channel that transmits zero bit of information about x, denoted by $p(z)$, and the encoder channel of VAEs, $q_\phi(z|x)$. Alemi et al. (2018) introduced the $H - D \leq \mathrm{I}(x;z) \leq R$ bounds[5], where H is the empirical data entropy (a constant). These bounds on mutual information allow us to analyze the trade-off between the reconstruction and KL terms in equation 5.1. For instance, since $\mathrm{I}(x;z)$ is non-negative (using Jensen's inequality), in the situation where $\mathrm{I}(x;z) = 0$, the encoder transmits no information about x, causing $R = 0, D = H$. Increasing $\mathrm{I}(x;z)$ can be encouraged by increasing both bounds: increasing the upper-bound (KL term) can be seen as the mean to control the maximum capacity of the encoder channel, while reducing the distortion (reconstruction loss) will tighten the bound by pushing the lower bound to its limits ($H - D \to H$). Similarly, channel capacity can be decreased.

[5]This is dependent on the choice of the encoder. For other bounds on mutual information see Johnson (2016); Poole et al. (2018).

5.2.2 Explicit KL Control via β-VAE

Given the above interpretation, we now turn to a slightly different formulation of ELBO based on β-VAE (Higgins et al., 2017). This allows control of the trade-off between the reconstruction and KL terms, as well as to set explicit KL value. While β-VAE offers regularizing the ELBO via an additional coefficient $\beta \in \mathbb{R}^+$, a simple extension (Burgess et al., 2018) of its objective function incorporates an additional hyperparameter C to explicitly control the magnitude of the KL term,

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta|D_{KL}\big(q_\phi(z|x)||p(z)\big) - C| \tag{5.2}$$

where $C \in \mathbb{R}^+$ and $|.|$ denotes the absolute value. While we could apply constraint optimization to impose the explicit constraint of KL$=C$, we found that the above objective function satisfies the constraint (Section 5.3). Alternatively, it has been shown (Pelsmaeker and Aziz, 2019) the similar effect could be reached by replacing the second term in equation 5.2 with $\max\big(C, D_{KL}\big(q_\phi(z|x)||p(z)\big)\big)$ at the risk of breaking the ELBO when KL$<C$ (Kingma et al., 2016).

5.3 Experiments

We conduct various experiments to illustrate the properties that are encouraged via different KL magnitudes. In particular, we start by revisiting the intrinsic properties of VAE: 1) the interdependence between rate and distortion, and 2) the impact of KL on the aggregated posterior (see Subsection 5.3.2) and approximate posterior. These two properties help us to understand the following experiments better. Then, through a set of qualitative and quantitative experiments for text generation, we demonstrate how certain generative behaviours could be imposed on VAEs via a range of maximum channel capacities. After that, we evaluate the discriminative performance of latent representations on three text classification tasks. Finally, we run some experiments to find if any form of syntactic information is encoded in the latent

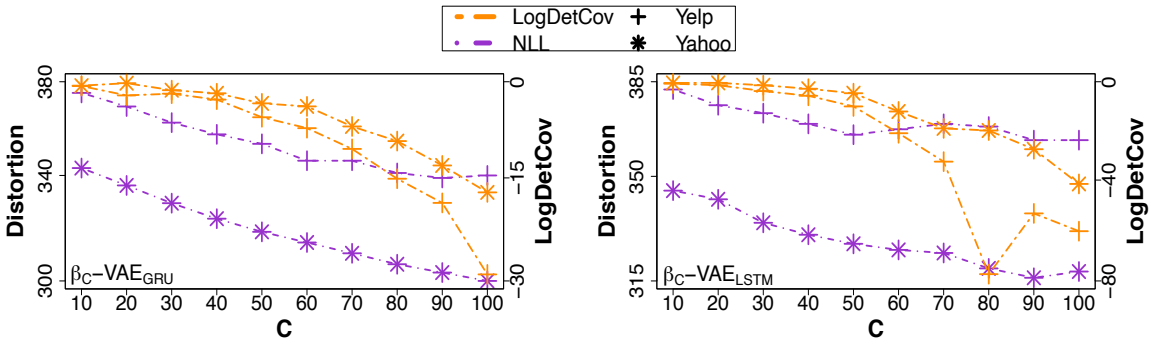


Fig. 5.1 Rate-Distortion and LogDetCov (see Subsection 5.3.2) for $C = \{10, 20, ..., 100\}$ on Yahoo and Yelp corpora.

space. For all experiments, we use the objective function of equation 5.2 with $\beta = 1$. We do not use larger βs because the constraint $\text{KL} = C$ is always satisfied with $\beta = 1$. [6, 7]

Corpora. We use 5 different corpora covering different domains and sizes through this section: Yelp and Yahoo (Yang et al., 2017) both have (100k,10k,10k) sentences in (train, dev, test) sets and 20k words in the vocabulary, Children's Book Test (CBT; Weston et al. (2016)) has (192k,10k,12k) sentences and 12k vocab, Wikipedia (WIKI; Marvin and Linzen (2018)) has (2m,270k,270k) sentences and 20k vocab, and WebText (Radford et al., 2019) has (1m,23k,24k) sentences and 22k vocab. Additionally, for the text classification experiment we use three corpora: Yelp, DBpedia and Yahoo. We use the same Yelp corpora as in the previous experiments, without any additional preprocessing. As for DBpedia[8] and Yahoo[9], the preprocessing is as follows: (1) removing all non-ASCII characters, quotations marks, and hyperlinks, (2) tokenising with spaCy[10], (3) lower-case conversion for all tokens, then (4) **for each class** we randomly sample 10,000 sentences for the training corpus and 1,000 sentences for the test and validation respectively. The vocabulary size of the both corpora is

[6] β can be seen as a Lagrange multiplier and any β value that allows for constraint satisfaction ($R = C$) is fine.

[7] Note, with values of $\beta < 1$ the constraint $\text{KL} = C$, potentially, may not be satisfied. We did not test this further as in our work we require $\text{KL} = C$ to be satisfied.

[8] https://github.com/srhrshr/torchDatasets/blob/master/dbpedia_csv.tar.gz

[9] https://github.com/jxhe/vae-lagging-encoder/blob/master/prepare_data.py.

[10] https://spacy.io

							Bucket 1		Bucket 2		All	
	C	D	R	LogDetCov	$\|\|\mu\|\|_2^2$	AU	BL2/RG2	BL4/RG4	BL2/RG2	BL4/RG4	BL2/RG2	BL4/RG4
CBT	3	62	3	-0.4	0.1	8	9.0/3.5	1.5/0.1	10.4/4.8	1.7/0.1	9.5/3.5	1.6/0.1
	15	53	15	-0.4	0.1	29	14.8/7.0	4.3/0.8	14.7/6.7	3.3/0.4	15.9/8.9	4.6/1.4
	100	32	99	-43.8	1.3	64	26.8/18.5	16.0/9.2	19.2/9.9	7.7/2.2	27.7/24.3	16.1/14.2
WIKI	3	81	3	-0.4	0.0	5	5.9/2.7	1.1/0.2	7.4/3.0	1.2/0.1	6.8/3.1	1.3/0.4
	15	70	15	-0.6	0.0	12	10.1/4.5	3.9/1.3	9.9/3.3	2.0/0.3	10.1/5.4	3.4/1.8
	100	17	100	-4.97	0.2	64	37.3/32.8	30.9/26.3	18.7/11.4	11.2/6.2	31.8/35.4	24.2/29.1
WebText	3	77	3	-0.2	0.0	4	9.6/4.6	1.7/0.2	12.6/6.4	4.0/1.0	11.9/5.5	3.4/0.7
	15	67	15	-0.5	0.0	16	15.5/7.4	5.4/1.5	15.6/7.3	5.6/1.6	15.8/7.9	5.7/1.8
	100	22	100	-7.9	0.4	64	61.7/58.3	56.4/53.1	35.1/27.3	27.31/21.0	45.8/45.3	38.7/39.7

Table 5.1 β_C-VAE$_{\text{LSTM}}$ performance with $C = \{3, 15, 100\}$ on the test sets of CBT, WIKI, and WebText. Each bucket groups sentences of certain length. Bucket 1: $10 <$ length ≤ 20; Bucket 2: $20 <$ length ≤ 30, and **All** contains all sentences of the corpus. BL2/RG2 denotes BLEU-2/ROUGE-2, BL4/RG4 denotes BLEU-2/ROUGE-2 BLEU-4/ROUGE-4, AU denotes active units, D denotes distortion and R denotes rate. For definition of LogDetCov and $||\mu||_2^2$ see Subsection 5.3.2.

reduced to the first 20,000 most frequent words. More information on the text classification corpora can be found in Appendix C.1.

Models. We examine two VAE architectures: β_C-VAE$_{\text{LSTM}}$ with (LSTM encoder, LSTM decoder) and β_C-VAE$_{\text{GRU}}$ with (GRU encoder (Cho et al., 2014b), GRU decoder). The dimension of word embeddings is 256 and the dimension of the latent variable is 64. The encoder and the decoder, for both VAE$_{\text{LSTM}}$ and VAE$_{\text{GRU}}$, have hidden size of 512 dimensions. Both models were trained for 10 epochs and optimised the objective function (equation 5.2) with Adam (Kingma and Ba, 2015) with the following learning rate: 8.5×10^{-4}.[11] To couple the encoder with the decoder we concatenate the latent variable to word embeddings at each time step without initialisation of hidden state.

[11] Learning rate and number of epochs: we use the vanilla VAE with the collapsed KL term to decide on the learning rate and the number of epochs. With the chosen, aforementioned, parameters the vanilla VAE has enough training iterations before it starts overfitting on the validation data.

5.3.1 Rate and Distortion

To analyse the dependence between the values of explicit rate (C) and distortion, we trained our models with different values of C, ranging from 10 to 100. Figure 5.1 reports the results for β_C-VAE$_{\text{GRU}}$ and β_C-VAE$_{\text{LSTM}}$ models on Yahoo and Yelp corpora. In all our experiments we found that $C-1 \leq KL \leq C+1$, demonstrating that the objective function effectively imposed the desired constraint on KL term.

The general trend is that by increasing the value of C one can get a better reconstruction (lower distortion) while the amount of gain varies depending on the VAE's architecture and corpus. [12] Additionally, we measured rate and distortion on CBT, WIKI, and WebText corpora using β_C-VAE$_{\text{LSTM}}$ and observed the same trend with the increase of C, see Table 5.1. This observation is consistent with the bound on $\mathrm{I}(x;z)$ we discussed earlier (Subsection 5.2.1) such that with an increase of KL we increase an upper bound on $\mathrm{I}(x;z)$ which in turn allows to have smaller values of reconstruction loss. Additionally, as reported in Table 5.1, encouraging higher rates (via larger C) encourages more active units (Burda et al., 2015, AU) in the latent code z.

As an additional verification, we also group the test sentences into buckets based on their length and report BLEU-2/4 and ROUGE-2/4 metrics to measure the quality of the reconstruction step in Table 5.1. As expected, we observe that increasing rate has a consistently positive impact on improving BLEU and ROUGE scores.

5.3.2 Impact of the Magnitude of KL Term on Aggregated Posterior and Approximate Posterior

During the text generation experiment (see Subsection 5.3.3) we generate samples from the prior $p(z)$ and in the text classification experiment we sample zs from $q(z|x)$. Hence we

[12] We attribute the difference in performance across our models to the non-optimal selection of training hyperparameters, and corpus-specific factors such as sentence length. Achieving SOTA results is not the goal of the experiment but rather show how change of C values influences the R for different architectures on neural networks and corpora.

would like to have a better understating of how the magnitude of the KL term can affect the approximate posterior and aggregated posterior, $q_\phi(z) = \sum_{x \sim q(x)} q_\phi(z|x)$, distributions.

For the text generation experiment, ideally, we would like the aggregated posterior to be as close as possible to the prior. This means that when we sample z from the prior distribution it will be in the same region that is covered by the approximate posterior distributions that we estimated for each point (sentence) in the training corpus. For the text classification experiment, we would expect better discriminative performance if there is a minimum overlap between the approximate posterior distributions - it would allow a classifier to better distinguish the sentences. This can be achieved if either the mean of the distributions are far apart or their shape is sharp - small standard deviation.

To understand how the approximate posterior are being affected by the magnitude of the KL, we adopted an approach from Zhao et al. (2017). We obtained unbiased samples of z first by sampling an x from data and then $z \sim q_\phi(z|x)$, and measured the log determinant of covariance (LogDetCov) of the samples ($\log\det(\mathrm{Cov}[q_\phi(z)])$). As reported in Figure 5.1, we observed that $\log\det(\mathrm{Cov}[q_\phi(z)])$ decreases as C grows, indicating sharper approximate posteriors.

We consider the difference of $p(z)$ and $q(z)$ in their means and variances, by computing the KL divergence from the moment-matching Gaussian fit of $q(z)$ to $p(z)$: This returns smaller values for $\beta_{C=5}$-VAE$_{\text{GRU}}$ (Yelp: 0, Yahoo: 0), and larger values for $\beta_{C=100}$-VAE$_{\text{GRU}}$ (Yelp: 8, Yahoo: 5), which illustrates that the overlap between $q_\phi(z)$ and $p(z)$ shrinks further as C grows. The above observation is better pronounced in Table 5.1, where we also report the mean ($||\mu||_2^2$) of unbiased samples of z, highlighting the divergence from the mean of the prior distribution as rate increases.

5.3.3 Text Generation

To empirically examine how the channel capacity translates into the generative capacity of the model, we experimented with the β_C-VAE$_{\text{LSTM}}$ models from Table 5.1. To generate a novel sentence, after a model was trained, a latent variable z is sampled from the prior distribution and then transformed into a sequence of words by the decoder $p_\theta(x|z)$.

During decoding for the generation we try three decoding schemes: (i) Greedy: which selects the most probable word at each step, (ii) Top-k (Fan et al., 2018): which at each step samples from the K most probable words, and (iii) Nucleus Sampling (Holtzman et al., 2019, NS): which at each step samples from a flexible subset of most probable words chosen based on their cumulative mass (set by a threshold p, where $p = 1$ means sampling from the full distribution). While similar to Top-k, the benefit of NS scheme is that the vocabulary size at each time step of decoding varies, a property that encourages diversity and avoids degenerate text patterns of greedy or beam search decoding (Holtzman et al., 2019). We experiment with NS ($p = \{0.5, 0.9\}$) and Top-k ($k = \{5, 15\}$).

Qualitative Analysis

We follow the settings of the homotopy experiment (Bowman et al., 2015b) where first a set of latent variables was obtained by performing a linear interpolation between $z_1 \sim p(z)$ and $z_2 \sim p(z)$. Then each z in the set was converted into a sequence of words by the decoder $p_\theta(x|z)$. Besides the initial motivation of Bowman et al. (2015b) to examine how neighbouring latent codes look like, our additional incentive is to analyse how sensitive the decoder is to small variations in the latent variable when trained with different channel capacities, $C = \{3, 15, 100\}$.

Table 5.2 shows the generated sentences via different decoding schemes for each channel capacity. Also, to make the generated sequences comparable across different decoding schemes or C values, we use the same samples of z for decoding. We only report the generated sentences for greedy, Top-$k = 15$, and NS $p = 0.9$. For the other values of k(=5) and p(=0.5) the generated sentences, under visual inspection, are of similar quality to the one we report.

Sensitivity of Decoder. To examine the sensitivity[13] of the decoder to variations of the latent variable, we consider the sentences generate with the greedy decoding scheme (the first column in Table 5.2). The other two schemes are not suitable for this analysis as they include

[13]Note: we vary z in one (randomly selected) direction (interpolating between z_1 and z_2). Alternatively, the sensitivity analysis can be done by varying z along the gradient direction of $\log p_\theta(x|z)$.

	Greedy	Top-15	NS(p=0.9)
C=3	1: oh, i m not going to be a good man.	1: come - look on my mind, said he.	1: and what is one of those trees creatures?
	2: oh, it s a good thing, said the story girl.	2: how could i tell you, that it s a great deal?	2: here s a nice heart among those waters!
	3: oh, how can you do it, dear?	3: said i. my sister, what a fool!	3: good-bye, said reddy fox, hardly frightened was out of his life.
	4: oh, how can you do it, dear?	4: and how was the way, you?	4: now, for a neighbor, who knows him.
	5: oh, how can you do it, miss?	5: said the other little breezes, but i do n t .	5: oh, prince ivan, dear me!
	6: and what is the matter with you?	6: and where s the news of the world?	6: cried her mother, who is hidden or power.
	7: and what is the matter with you?	7: ⟨unk⟩ of ⟨unk⟩, said i. ay, ⟨unk⟩!	7: but this was his plight, and the smith knew.
C=15	1: old mother west wind and her eyes were in the same place, but she had never seen her.	1: eric found out this little while, but there in which the old man did not see it so.	1: aunt tommy took a sudden notion of relief and yellow-dog between him sharply until he tried to go to.
	2: old mother west wind and his wife had gone and went to bed to the palace.	2: old mother west wind and his wife gave her to take a great ⟨unk⟩, she said.	2: his lord marquis of laughter expressed that soft hope and miss cornelia was not comforted.
	3: little joe otter and there were a ⟨unk⟩ of them to be seen.	3: little joe otter got back to school all the ⟨unk⟩ together.	3: meanwhile the hounds were both around and then by a thing was not yet.
	4: little joe otter s eyes are just as big as her.	4: little joyce s eyes grew well at once, there.	4: in a tone, he began to enter after dinner.
	5: a few minutes did not answer the ⟨unk⟩.	5: pretty a woman, but there had vanished.	5: once a word became, just got his way.
	6: a little while they went on.	6: from the third day, she went.	6: for a few moments, began to find.
	7: a little while they went.	7: three months were as usual.	7: meantime the thrushes were ⟨unk⟩.
C=100	1: it will it, all her ⟨unk⟩, not even her with her?	1: it will her you, at last, bad and never in her eyes.	1: it s; they liked the red, but i kept her and growing.
	2: it will get him to mrs. matilda and nothing to eat her long clothes.	2: other time, i went into a moment – she went in home and.	2: it ⟨unk⟩ not to her, in school, and never his bitter now.
	3: the thing she put to his love, when it were ⟨unk⟩ and too.	3: going quite well to his mother, and remember it the night in night!	3: was it now of the beginning, and dr. hamilton was her away and.
	4: one day, to the green forest now and a long time ago, sighed.	4: one and it rained for his feet, for she was their eyes like ever.	4: of course she flew for a long distance; and they came a longing now.
	5: one and it became clear of him on that direction by the night ago.	5: the thing knew the tracks of ⟨unk⟩ and he never got an ⟨unk⟩ before him.	5: one door what made the pain called for her first ear for losing up.
	6: every word of his horse was and the rest as the others were ready for him.	6: of course he heard a sound of her as much over the ⟨unk⟩ that night can.	6: one and he got by looking quite like her part till the marriage know ended.
	7: a time and was half the ⟨unk⟩ as before the first ⟨unk⟩ things were ready as.	7: every, who had an interest in that till his legs got splendid tongue than himself.	7: without the thought that danced in the ground which made these delicate child s teeth so.

Table 5.2 Homotopy (CBT corpus) - The three blocks correspond to $C = \{3, 15, 100\}$ values used for training β_C-VAE$_{\text{LSTM}}$. The columns correspond to the three decoding schemes: greedy, top-k (with k=15), and the nucleus sampling (NS; with p=0.9). Initial two latent variables z were sampled from a the prior distribution i.e. $z \sim p(z)$ and the other five latent variables were obtained by interpolation. The sequences that highlighted in gray are the one that decoded into the same sentences condition on different latent variable. **Note**: Even though the learned latent representation should be quite different for different models (trained with different C) in order to be consistent all the generated sequences presented in the table were decoded from the same seven latent variables.

sampling procedure. This means that if we decode the same latent variable twice we will get two different sentences. We observed that with lower channel capacity ($C = 3$) the decoder tends to generate identical sentences for the interpolated latent variables (we highlight these sentences in gray), exhibiting decoder's lower sensitivity to z's variations. However, with the increase of channel capacity ($C = 15, 100$) the decoder becomes more sensitive. This observation is further supported by the increasing pattern of active units in Table 5.1: Given that AU increases with the increase of C one would expect that the activation pattern of a latent variable becomes more complex as it comprises more information. Therefore a small change in the pattern would have a greater effect on the decoder.

Coherence of Sequences. We observe that the model trained with large values of C compromises sequences' coherence during the sampling. This is especially evident when we compare $C = 3$ with $C = 100$. Analysis of Top-15 and NS (p=0.9) generated samples reveals that the lack of coherence is not due to the greedy decoding scheme per se, and can be attributed to the model in general. To understand this behavior further, we need two additional results from Table 5.1: LogDetCov and $||\mu||_2^2$. One can notice that as C increases LogDetCov decreases and $||\mu||_2^2$ increases. This indicates that the aggregated posterior becomes further apart from the prior, hence the latent codes seen during the training diverge more from the codes sampled from the prior during generation. We speculate this contributes to the coherence of the generated samples, as the decoder is not equipped to decode prior samples properly at higher Cs.

Quantitative Analysis

Quantitative analysis of generated text without gold reference sequences (e.g. in Machine Translation or Summarization) has been a long-standing challenge. Recently, there have been efforts towards this direction, with proposal such as self-BLEU (Zhu et al.), forward cross entropy (Cífka et al., 2018, FCE) and Fréchet InferSent Distance (Cífka et al., 2018, FID). We opted for FCE as a complementary metric to our qualitative analysis.

To calculate FCE, first a collection of synthetic sentences are generated by sampling $z \sim p(z)$ and decoding the samples into sentences. The synthetic sequences are then used to train a language model (an LSTM with the parametrisation of our decoder). The FCE score is estimated by reporting the negative log likelihood (NLL) of the trained LM on the set of human-generated sentences.

We generated synthetic corpora using trained models from Table 5.1 with different C and decoding schemes and using the same exact z samples for all corpora. Since the generated corpora using different C values would have different coverage of words in the test set (i.e., Out-of-Vocabulary ratios), we used a fixed vocabulary to minimize the effect of different vocabularies in our analysis. Our dictionary contains words that are common in all of the three corpora, while the rest of the words that don't exist in this dictionary are replaced with $\langle$unk$\rangle$ symbol. Similarly, we used this fixed dictionary to preprocess the test sets. Also, to reduce bias to a particular set of sampled z's we measure the FCE score three times, each time we sampled a new training corpus from a β_C-VAE$_{\text{LSTM}}$ decoder and trained an LM from scratch. In Table 5.3 we report the average FCE (NLL) for the generated corpora.

In the qualitative analysis we observed that the text generated by the β_C-VAE$_{\text{LSTM}}$ trained with large values of $C = 100$ exhibits lower quality (i.e., in terms of coherence). This observation is supported by the FCE score of NS(p=0.9) decoding scheme (Table 5.3), since the performance drops when the LM is trained on the corpus generated with $C = 100$. The generated corpora with $C = 3$ and $C = 15$ achieve similar FCE score. However, these patterns are reversed for Greedy decoding scheme[14], where the general tendency of FCE scores suggests that for larger values of C the β_C-VAE$_{\text{LSTM}}$ seems to generate text which better approximates the natural sentences in the test set. To understand this further, we report additional statistics in Table 5.3: percentage of $\langle$unk$\rangle$ symbols, self-BLEU and average sentence length in the corpus.

The average sentence length, in the generated corpora is very similar for both decoding schemes, removing the possibility that the pathological pattern on FCE scores was caused by difference in sentence length. However, we observe that for Greedy decoding more

[14]For the other decoding schemes: Top-{5,15} and NS(p=0.5) the pattern is the same as for the Greedy.

		Greedy					NS(p=0.9)				
	C	|V|	FCE ↓	%unk	len.	SB ↓	|V|	FCE ↓	%unk	len.	SB ↓
CBT	3	335	86.6(0.4)	9.7	15.3	4.2	9.8k	**70.4(0.0)**	2.1	15.6	0.0
	15	335	52.3(0.3)	12.7	15.2	0.3	9.8k	70.7(0.2)	2.4	15.4	0.0
	100	335	**47.3(0.1)**	21.3	17.5	0.0	9.8k	75.1(0.1)	2.2	17.6	0.0
Test		328	-	30.7	15.3	-	6.1k	-	3.6	15.3	-
WIKI	3	1.5k	134.6(0.8)	27.3	19.9	7.6	20k	89.8(0.1)	5.8	19.4	0.0
	15	1.5k	69.2(0.1)	18.9	19.8	0.2	20k	**89.3(0.1)**	5.6	19.8	0.0
	100	1.5k	**58.9(0.1)**	34.8	20.7	0.0	20k	96.5(0.1)	4.5	20.7	0.0
Test		1.5k	-	32.7	19.6	-	20k	-	5.2	19.6	-
WebText	3	2.3k	115.8(0.7)	18.8	17.5	2.0	21.9k	86.4(0.1)	7.1	15.6	0.0
	15	2.3k	74.4(0.1)	15.5	15.8	0.1	21.9k	**85.8(0.1)**	6.9	15.9	0.0
	100	2.3k	**62.5(0.1)**	27.3	18.0	0.0	21.9k	93.7(0.1)	4.8	18.0	0.0
Test		2.2k	-	30.1	16.1	-	17.1k	-	6.8	16.1	-

Table 5.3 Forward Cross Entropy (FCE). Columns represent stats for Greedy and NS decoding schemes for β_C-VAE$_{\text{LSTM}}$ models trained with $C = \{3, 15, 100\}$ on CBT, WIKI or WebText. Each entry in the table is a mean of negative log likelihood of an LM. The values in the brackets are the standard deviations. |V| is the vocabulary size; Test stands for test set; %unk is the percentage of $\langle$unk$\rangle$ symbols in a corpora; len. is the average length of a sentence in the generated corpus; SB is the self-BLEU:4 score calculated on the 10K sentences in the generated corpus.

than 30% of the test set consists of $\langle$unk$\rangle$. Intuitively, seeing more evidence of this symbol during training would improve our estimate for the $\langle$unk$\rangle$. As reported in the table, the %unk increases on almost all corpora as C grows, which is then translated into getting a better FCE score at test. Therefore, we believe that FCE at high %unk is not a reliable quantitative metric to assess the quality of the generated syntactic corpora. Furthermore, for Greedy decoding, self-BLEU decreases when C increases. This suggests that generated sentences for higher value of C are more diverse. Hence, the LM trained on more diverse corpora can generalise better, which in turn affects the FCE.

In contrast, the effect the $\langle$unk$\rangle$ symbol has on the corpora generated with the NS(p=0.9) decoding scheme is minimal for two reasons: First, the vocabulary size for the generated corpora, for all values of C is close to the original corpus (the corpus we used to train the β_C-VAE$_{\text{LSTM}}$). Second, the vocabularies of the corpora generated with three values of C is very close to each other. As a result, minimum replacement of the words with the $\langle$unk$\rangle$ symbol is required, making the experiment to be more reflective of the quality of the

generated text. Similarly, self-BLEU for the NS(p=0.9) is the same for all values of C. This suggests that the diversity of sentences has minimal, if any, effect on the FCE.

5.3.4 Text Classification

Prior to use of a VAE encoder in the classification experiment, we pretrain it using the full VAE model with the VAE's objective function (equation 5.2) using one of the following C values: 0, 3, 15 and 100. We train each of the VAE models on the three text classification corpora: Yelp, Yahoo or DBpedia. Furthermore, we experiment with the two VAE architectures: β_C-VAE$_{\text{GRU}}$ and β_C-VAE$_{\text{LSTM}}$.

To train the classifier, $p(y|x)$, with a probabilistic VAE encoder we marginalise the latent variable(s). This is done as follows, given the classifier:

$$p(y_i|x_i) = \int_z p(y_i|z)q_\phi(z|x_i)dz,$$

where the (y_i, x_i) is a single input/output pair in the corpus, we use Monte Carlo (MC) approximation to estimate the integral of the classifier. We approximate the integral by taking five samples from the probabilistic encoder both to train and to test the classifier: For each x_i in a batch $\{x_1, ..., x_p\}$ sample five of $z_{i,j}$ from $q_\phi(z|x_i)$ i.e. a set of sampled z's is $\{z_{i,1}, ..., z_{i,5}\}$. With the MC approximation: $p(y_i|x_i) \approx 0.2 \times \sum_{j=1}^{5} p(y_i|z_{i,j})$.

In Table 5.4, we compare the performance of the VAE models trained with the various values of C on the three text classification tasks. To establish whether the performance gain or loss on the tasks is achieved thanks to the information-theoretic inductive bias, for all the VAE models we freeze the parameters of the encoder and only train the classifier[15] which we put on top of the encoder.

The general trend that we observe is that with the increase of the value of C the classification performance is increasing. We attribute this to both the narrow approximate posterior distribution, which allows the classifier to better distinct between the points, and the increased

[15]The classifier comprises of the feedforward neural networks (with dense or fully connected layers). The first two layers are of 32 dimensions each with LekyReLU activation functions. The final layer of the classifier has softmax activation function and the number of its dimensions is equal to the number of classes.

Models	Yelp			DBPedia			Yahoo		
	Acc. ↑	KL. ↑	R ↓	Acc. ↑	KL ↑	R ↓	Acc. ↑	KL ↑	R ↓
$\beta_{C=0}$-VAE$_{\text{GRU}}$	0.2 ± 0.0	0.0 ± 0.0	386.3 ± 1.3	0.1 ± 0.0	0.0 ± 0.0	113.0 ± 0.0	0.1 ± 0.0	0.0 ± 0.0	58.0 ± 0.0
$\beta_{C=0}$-VAE$_{\text{LSTM}}$	0.2 ± 0.0	0.0 ± 0.0	392.0 ± 0.8	0.1 ± 0.0	0.0 ± 0.0	113.0 ± 0.0	0.1 ± 0.0	0.0 ± 0.0	57.0 ± 0.0
$\beta_{C=3}$-VAE$_{\text{GRU}}$	0.5 ± 0.0	3.3 ± 0.5	382.7 ± 0.9	0.4 ± 0.0	3.0 ± 0.0	110.0 ± 0.0	0.2 ± 0.0	3.0 ± 0.0	55.0 ± 0.0
$\beta_{C=15}$-VAE$_{\text{GRU}}$	0.5 ± 0.0	15.0 ± 0.0	373.0 ± 0.0	0.8 ± 0.0	15.0 ± 0.0	101.0 ± 0.0	0.3 ± 0.0	15.7 ± 0.5	45.0 ± 0.0
$\beta_{C=100}$-VAE$_{\text{GRU}}$	**0.6 ± 0.0**	100.0 ± 1.4	336.0 ± 0.0	**0.9 ± 0.0**	99.7 ± 0.5	81.0 ± 0.0	0.4 ± 0.0	99.7 ± 0.9	25.7 ± 0.5
$\beta_{C=3}$-VAE$_{\text{LSTM}}$	0.5 ± 0.0	3.0 ± 0.0	389.0 ± 0.0	0.6 ± 0.0	3.0 ± 0.0	110.0 ± 0.0	0.2 ± 0.0	3.0 ± 0.0	54.3 ± 0.5
$\beta_{C=15}$-VAE$_{\text{LSTM}}$	0.5 ± 0.0	15.0 ± 0.0	381.3 ± 1.3	0.7 ± 0.1	15.0 ± 0.0	104.7 ± 3.8	0.3 ± 0.0	15.0 ± 0.0	45.0 ± 0.0
$\beta_{C=100}$-VAE$_{\text{LSTM}}$	0.5 ± 0.0	100.0 ± 0.8	373.3 ± 4.5	0.7 ± 0.2	100.0 ± 0.0	101.7 ± 4.8	0.4 ± 0.0	99.3 ± 0.5	30.3 ± 0.5

Table 5.4 The reconstruction loss (R), Kullback-Leibler term (KL) and the classification accuracy (Acc.) for the VAEs evaluated on the corresponding test corpus. We train each VAE model three times; in the table we report the mean and the standard deviation of R, KL and Acc. over the three runs of the models. The latent code of the VAEs is 64 dimensions. The weights of the VAE encoders are frozen during the training of the classifiers.

amount of information that is stored in the sentence embeddings. Also, as we discussed in Subsection 5.3.1, the gain we get by increasing the value of C depends on VAE architecture. In our experiments, β_C-VAE$_{\text{GRU}}$ benefits more than β_C-VAE$_{\text{LSTM}}$ from larger value of Cs. We hypothesis that could be due to the non-optimal[16] selection of training hyperparameters that we use to train the β_C-VAE$_{\text{LSTM}}$. We leave this to future investigation.

5.3.5 Syntactic Test

In this section, we explore if any form of syntactic information is captured by the encoder and represented in the latent codes despite the lack of any explicit syntactic signal during the training of the β_C-VAE$_{\text{LSTM}}$. To train the models we used the same WIKI data set as in Marvin and Linzen (2018), but we filtered out all the sentences that are longer than 50 space-separated tokens.[17]

We use the data set constructed by Marvin and Linzen (2018), which comprises of 337,072 pairs of grammatical and ungrammatical English sentences,[18] to test three syntactic phenomena: subject-verb agreemnet, reflexive anaphora and negative polarity items. For

[16] Different learning rate, initialisation of weights of LSTM neural networks and different ways of coupling of encoder and decoder can potentially improve the performance.

[17] We applied the filtering to decrease the training time of our models.

[18] The sentences were constructed using the templates which are based on nonrecursive context-free grammars.

	$C=3$				$C=100$			
Syntactic Categories	p_1	p_2	$\bar{p}_1$	$\bar{p}_2$	p_1	p_2	$\bar{p}_1$	$\bar{p}_2$
SUBJECT-VERB AGREEMENT								
Simple	0.81	0.81	0.81	0.81	1.0	0.23	0.68	0.47
In a sentential complement	0.79	0.79	0.79	0.79	0.98	0.14	0.69	0.48
Short VP coordination	0.74	0.73	0.73	0.73	0.96	0.08	0.78	0.43
Long VP coordination	0.61	0.61	0.60	0.60	0.97	0.06	0.55	0.47
Across a prepositional phrase	0.78	0.78	0.82	0.82	0.97	0.07	0.62	0.49
Across a subject relative clause	0.77	0.77	0.77	0.77	0.93	0.08	0.68	0.41
Across an object relative clause	0.69	0.69	0.69	0.69	0.92	0.11	0.61	0.45
Across an object relative (no that)	0.58	0.58	0.58	0.58	0.94	0.09	0.61	0.44
In an object relative clause	0.74	0.74	0.74	0.74	0.99	0.01	0.60	0.45
In an object relative (no that)	0.74	0.74	0.74	0.74	0.99	0.02	0.61	0.46
REFLEXIVE ANAPHORA								
Simple	0.79	0.78	0.80	0.79	0.99	0.07	0.70	0.39
In a sentential complement	0.74	0.73	0.74	0.73	1.00	0.00	0.70	0.38
Across a relative clause	0.63	0.62	0.63	0.62	0.99	0.03	0.69	0.35
NEGATIVE POLARITY ITEMS								
Simple	0.42	0.33	0.41	0.37	1.00	0.00	0.76	0.20
Across a relative clause	0.37	0.36	0.35	0.34	1.00	0.00	0.98	0.02

Table 5.5 p_1: $p(x^-|z^+) < p(x^+|z^+)$ and p_2: $p(x^-|z^-) < p(x^+|z^-)$; $\bar{p}_1$: $p(x^-|\bar{z}^+) < p(x^+|\bar{z}^+)$ and $\bar{p}_2$: $p(x^-|\bar{z}^-) < p(x^+|\bar{z}^-)$; $\beta_{C=3}$-VAE$_{\text{LSTM}}$ (D:103, R:3); $\beta_{C=100}$-VAE$_{\text{LSTM}}$ (D:39, R:101).

example, a pair in subject-verb agreement category would be: (*The author laughs*, *The author laugh*).

We encode both the grammatical and ungrammatical sentences into the latent codes z^+ and z^-, respectively. Then we condition the decoder on the z^+ and try to determine whether the decoder assigns a higher probability to the grammatical sentence (denoted by x^+): $p(x^-|z^+) < p(x^+|z^+)$ (denoted by p_1 in Table 5.5). We repeat the same experiment but this time try to determine whether the decoder, when conditioned on the ungrammatical code (z^-), still prefers to assign higher probability to the grammatical sentence: $p(x^-|z^-) < p(x^+|z^-)$ (denoted by p_2 in Table 5.5). Table 5.5 shows the p_1 and p_2 for the β_C-VAE$_{\text{LSTM}}$ model trained with $C = \{3, 100\}$. Both the p_1 and p_2 are similar to the accuracy and correspond to how many times a grammatical sentence was assigned a higher probability.

As reported for C=3, p_1 and p_2 match in almost all cases. This is to some degree expected since the dependence of the decoder on the latent code is so negligible that the decoder hardly distinguishes the grammatical and ungrammatical inputs. This changes for $C = 100$,

as in almost all the cases the decoder becomes strongly dependent on the latent code and can differentiate between what it has seen as input and the closely similar sentence it hasn't received as the input: the decoder assigns larger probability to the ungrammatical sentence when conditioned on the z^- and, similarly, larger probability to the grammatical sentence when conditioned on the z^+.

However, the above observations neither confirm nor reject existence of a grammar signal in the latent codes. We run a second set of experiments where we aim to discard sentence specific information from the latent codes by averaging the codes[19] inside each syntactic category. The averaged codes are denoted by $\bar{z}^+$ and $\bar{z}^-$, and the corresponding accuracies are reported by $\bar{\text{p}}_1$ and $\bar{\text{p}}_2$ in Table 5.5. Our hypothesis is that the only invariant factor during averaging the codes inside a category is the grammatical property of its corresponding sentences.

As expected, due to the weak dependence of the decoder on the latent code, the performance of the model under $C = 3$ is almost identical when comparing p_1 vs. $\bar{\text{p}}_1$, and p_2 vs. $\bar{\text{p}}_2$. However, for $C = 100$ the performance of the model deteriorates. We leave further exploration of this behavior to our future work.

5.4 Conclusion

In this chapter we used an information-theoretic inductive bias, formulated within a VAE model, to control the amount of information transmitted between the encoder and decoder via the sentence embedding z. We control the amount of information via the KL term. To study the implications this inductive bias has on components (generative and inference networks) of the VAE model we used downstream tasks.

First, we tested the impact this bias has on the generative capacity of the VAE model. We showed that small and large values of the KL term impose different properties on the generated text: the decoder trained under the smaller KL term tends to generate repetitive but mainly plausible sentences, while for larger KL the generated sentences were diverse

[19] Each syntactic category is further divided into sub-categories, for instance *simple subject-verb agreement* We average z's within each sub-categories.

but incoherent. This behaviour was observed across three different decoding schemes and complemented by a quantitative analysis where we measured the performance of an LSTM LM trained on different VAE-generated synthetic corpora via different KL magnitudes, and tested on human-generated sentences.

Then, we analysed how the bias affects the encoder network. We used three text classification tasks for this. We demonstrated that sentence embeddings that store more amount of information are superior, performance (accuracy) wise, on the tasks compared to sentence embeddings that store less amount of information. This means that the encoder network infers more representative latent representations of the sentences.

Finally, using a language modeling task, we attempted to understand if the bias allows the model to distinguish between grammatical and ungrammatical sentences. By increasing the amount of information, in order to better represent the sentence in the latent code, the model may decide to encode syntactic information into the sentence embeddings. We verified that at lower (and still non-zero) KL the decoder tends to pay less attention to the latent code, but our findings regarding the presence of a syntactic signal in the latent code were inconclusive. We leave it as a possible avenue to explore in our future work.

In the next chapter, we built on our VAE framework and explore sparsity inductive bias. More concretely, we discuss how sparse representations could be a more natural way of modeling sentences in a fixed dimensional vector.

6

Learning Sentence Embeddings with Sparsity Inductive Bias

6.1 Introduction[1]

Representation learning has been pivotal in many success stories of modern days NLP. Observing its success, two fundamental questions arise: *How is the information encoded in them?* and *What is encoded in them?* While the latter has received a lot of attention by designing probing tasks, the former has been largely neglected.

In this chapter, we take small steps in this non-trivial direction by building on the knowns: One property we know about the encoding of information is that different data points embody different characteristics (e.g. statistically, semantically, or syntactically) which should ideally utilise different sub-regions of the representation space. Therefore, the high-dimensional learned representations should ideally be sparse (Bengio et al., 2013; Burgess et al., 2018; Tonolini et al., 2019) to have varying number of active dimension per sentence (Bengio, 2009) in a fixed dimensional vector.[2] But *if sparsity is expected, could it be learned from data without supervision?* We investigate the answer to this question with a sparsity inductive bias, incorporated into a model.

[1]This chapter draws from the following publication: **Victor Prokhorov**, Yingzhen Li, Ehsan Shareghi and Nigel Collier (RepL4NLP, 2021) "Learning Sparse Sentence Encoding without Supervision: An Exploration of Sparsity in Variational Autoencoders".

[2]More on speculative side, sparse representations may be a more natural way of modelling sentences of a language in a fixed dimensional vector. Sentences vary in length and the amount of information they convey. As such it makes sense to reflect this property in a vector representation of the sentence.

A handful of studies in NLP that have delved into building sparse representations of words either during the learning phase (Faruqui and Dyer, 2015; Yogatama et al., 2015) or as a post-processing step on top of existing representations (e.g., word2vec embeddings) (Faruqui et al., 2015; Sun et al., 2016; Arora et al., 2018; Subramanian et al., 2018a; Li and Hao, 2019). These methods have not been developed for sentence embeddings, with the exception of Trifonov et al. (2018) which makes a strong assumption by forcing the latent sentence representation to be a sparse categorical distribution.

In parallel, Variational Autoencoders (VAEs; Kingma and Welling (2014)) have been effective in capturing semantic closeness of sentences in the learned representation space (Bowman et al., 2015b; Prokhorov et al., 2019; Xu et al., 2019; Balasubramanian et al., 2020). Furthermore, methods have been developed to encourage sparsity in VAEs via learning a deterministic selection variable (Yeung et al., 2017) or sparse priors (Barello et al., 2018; Mathieu et al., 2019; Tonolini et al., 2019). However, the success of these is yet to be examined on text domain.

To bridge this gap, we make a sober evaluation of existing state-of-the-art (SOTA) VAE-based sparsification model (Mathieu et al., 2019) against several VAE-based baselines on two experimental tasks: text classification accuracy, and the level of representation sparsity achieved. Additionally, we propose Hierarchical Sparse Variation Autoencoder (HSVAE), to improve the stability issue of existing SOTA model[3] and demonstrate its performance on both experimental tasks.[4]

Our experimental findings demonstrate that: (I) neither the simpler baseline models nor the SOTA manage to impose a satisfactory level of sparsity on text, (II) as expected, sparsity level and task performance have a negative correlation, while giving up task performance and having sparse codes helps with the analysis of the representations, (III) presence/absence of task related signal in the sparsity codes affects the task performance, (IV) the success of capturing the task related signal in the sparsity codes depends on the strength of the

[3]Please refer to Appendix D.2 to further understand the difference between the two VAEs; HSVAE and MAT-VAE.

[4]As in Mathieu et al. (2019), we induce sparse representations for each data point, also known as ephemeral sparsity (Hoefler et al., 2021).

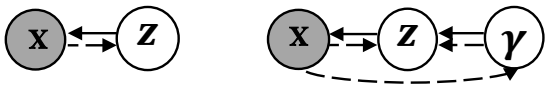


Fig. 6.1 Graphical Models of VAE (left) and HSVAE (right). Solid and dashed lines represent generative and inference paths, respectively.

signal presented in a corpus, and representation dimensionality, (V) the success of SOTA in image domain does not necessarily transfer to inducing sparse representations for text, while HSVAE addresses this shortcoming.

6.2 Hierarchical Sparse VAE (HSVAE)

We propose the hierarchical sparse VAE (HSVAE), Figure 6.1 (right), to learn sparse latent codes automatically. We treat the mixture weights $\gamma = (\gamma_1, ..., \gamma_D)$ as a random variable and assign a factorised Beta prior $p_\theta(\gamma_i) = \text{Beta}(\alpha, \beta)$ on it. The latent code z is then sampled from a factorised Spike-and-Slab distribution[5] (Mitchell and Beauchamp, 1988; Ishwaran and Rao, 2005) $p_\theta(z|\gamma)$ conditioned on γ, and the observation x is generated by decoding the latent variable $x \sim p_\theta(x|z)$ using a GRU (Cho et al., 2014b) decoder. This returns a probabilistic generative model $p_\theta(x,z,\gamma) = p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)$. For posterior inference, the encoder distribution is defined as $q_\phi(z,\gamma|x) = q_\phi(\gamma|x)q_\phi(z|\gamma,x)$, where $q_\phi(\gamma|x)$ is a learnable and factorised Beta distribution, and $q_\phi(z|\gamma,x)$ is a factorised Spike-and-Slab distribution with mixture weights γ_i and learnable "slab" components for each dimension. The q distribution is computed by first extracting features from the sequence using a GRU, then applying MLPs to the extracted feature (and γ for $q_\phi(z|\gamma,x)$) to produce the distributional parameters.

ELBO. We derive the ELBO, $\mathscr{L}(\theta,\phi;x)$:

[5]This is a mixture of two Gaussians with mixture weight γ_i, where the *slab* component is a standard Gaussian while the *spike* component is a Gaussian with $\sigma \to 0$:

$$p(z) = \prod_i^{\text{D}} (1-\gamma_i)\,N(z_i;0,1) + \gamma_i\,N(z_i;0,\sigma \to 0)$$

where i denotes the ith dimension of z and D is the total number of dimensions of z.

$$\mathbb{E}_{q_\phi(z,\gamma|x)}[\log p_\theta(x|z)] - \psi\mathbb{E}_{q_\phi(\gamma|x)}[\mathbb{D}_{KL}\big(q_\phi(z|\gamma,x), p_\theta(z|\gamma)\big)] - \lambda\mathbb{D}_{KL}\big(q_\phi(\gamma|x)||p_\theta(\gamma)\big),$$

where $\psi \in \mathbb{R}$ and $\lambda \in \mathbb{R}$ are the coefficients for the KL terms. This ELBO (see Appendix D.1 for ELBO derivation) is approximated with Monte Carlo (MC) in practice, $\mathscr{L}(\theta,\phi;x)$:

$$\frac{1}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\left[\frac{1}{M}\sum_{z\sim q_\phi(z|x,\gamma)}^{M}\log p_\theta(x|z)\right] - \frac{\psi}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\Big[\mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma))\Big] - \\ -\lambda\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)),$$

where M and N are scalar numbers corresponding to a number of samples taken from $q_\phi(z|x,\gamma)$ and $q_\phi(\gamma|x)$ respectively. In this work, we set both M and N to 1. Similar to the vanilla VAE, the first term is the reconstruction, the second and the third KL terms[6] control the distance between the posteriors and their corresponding priors. The parameters of the priors are fixed to some constant values (can be also thought as the hyperparameters) during the training.

Control of Sparsity. The random variable γ_i, in our model, can be viewed as a "probabilistic switch" that determines how likely is for the ith dimension of z to be turned off. Intuitively, since for both generation and inference the latent code z is sampled from a Spike-and-Slab distribution with the mixture weights γ, $\gamma_i \to 1$ means z_i is drawn from a delta mass centered at $z_i = 0$. As the switch follows a Beta distribution $\gamma_i \sim Beta(\gamma_i;\alpha,\beta)$, we can select the parameters α and β to control the concentration of the probability mass on $\gamma_i \in [0,1]$ interval.

There are three typical configurations of the (α,β) pair: (1) $\alpha < \beta$: density is shifted towards $\gamma_i = 0$ hence ith unit is likely to be on and dense representation is expected, (2) $\alpha = \beta$: the density is centered at $\gamma_i = 0.5$, and (3) $\alpha > \beta$: density is shifted towards $\gamma_i = 1$, hence the unit is likely to be off, leading to sparsity. The magnitude of these parameters also plays a role as it controls the spread and uni/bi-modal structure of the density.

[6]The first KL term is estimated via MC and the the second KL term is calculated in the closed form.

6.3 Experiments

As in Chapter 5, we conduct a set of experiments on three text classification corpora (see Appendix C.1): Yelp (sentiment analysis - 5 classes) (Yang et al., 2017), DBpedia and Yahoo (topic classification - 14 and 10 classes respectively) (Zhang et al., 2015). First, we compare performance of the sparse latent representations with their dense counterpart on the text classification tasks (Subsection 6.3.2). Second, the stability of sparsification of HSVAE is compared with the state-of-the-art MAT-VAE (Subsection 6.3.3). Then, to better understand performance of our model on the downstream task, we examine the sparsity patterns (Subsection 6.3.4).

An integral part of the experiments is the analysis of the learned representations. In this sense, tasks that rely on understanding of semantics (e.g., GLUE Wang et al. (2018)) or syntax (e.g., Marvin and Linzen (2018)) would be non-trivial to analyse due to their inherent complexity. We consider classification tasks because the distribution of words alone could be a good indicator of class labels. Given the unsupervised nature of the models, we explore if this surface-level distribution of words could be captured by the sparsity patterns in the learned representation.

6.3.1 Experimental Setup

Baselines and Models

To ground the performance of HSVAE we use 4 baselines: 1) VAE is a version of the vanilla VAE used in Higgins et al. (2017), 2) the same VAE model but the activation of μ and σ of $q_\phi(z|x)$ regularised by either L^1 (VAE$_{L^1}$) or L^2 (VAE$_{L^2}$) norms, 3) MAT-VAE is a VAE framework introduced by Mathieu et al. (2019) and 4) simple classifier which is simply a text encoder with a classifier on top of it. For all these models we use a GRU network to encode and decode text sequences. We set the dimesnionality of the both encoder and the decoder GRU's to 512D and the dimensionality of the word embeddings is 256D. The decoder and

the encoder share the word embeddings. To train the model we use the Adam optimiser (Kingma and Ba, 2015) with the learning rate: 0.0008.[7]

BERT vs GRU Encoder. Inspired by Li et al. (2020b), we replace the GRU network used in VAE and HSVAE encoders with a pretrained BERT[8] (Devlin et al., 2019), while keeping the GRU decoder. We refer to these models as B-VAE and B-HSVAE, respectively. Also, we compare the task performance of these VAE models with the plain pretrained base-BERT.[9] To train B-VAE and B-HSVAE, we use the Adam optimiser with the learning rate: 0.00008.[10]

Dimensionality of z. We use the following two dimensions: 32D and 768D. Since, HSVAE and MAT-VAE induce sparse latent representations we want to make sure that they perform robustly regardless of the number of the dimensions.

KL-Collapse. None of the used VAE models is immune to the KL-collapse (Bowman et al., 2015b) - when the KL term becomes zero and the decoder ignores the information provided by the encoder through z. To address this issue, in all the models, we put a scalar value $\psi, \lambda < 1$ on the KL terms of the VAE's objective function (He et al., 2019).

Coupling Encoder with Decoder. To connect the encoder with the decoder we concatenate the latent variable z, sampled from the posterior distribution, to word embeddings of the decoder at each time step (Prokhorov et al., 2019). Also, for GRU encoders we take the last hidden state to parameterise the posterior distribution. For BERT encoder, we take average pooling of all token's embeddings produced by the last layer of BERT.

[7] Learning rate and number of epochs: we use the vanilla VAE with the collapsed KL term to decide on the learning rate and the number of epochs. With the chosen, aforementioned, parameters the vanilla VAE has enough training iterations before it starts overfitting on the validation data. We train the models for 15 epochs.

[8] After extracting features from a sequence with BERT, we then applying MLPs to extract features for the posterior distributions, as it is the case for the encoder with GRU network.

[9] https://huggingface.co/transformers/model_doc/bert.html

[10] We empirically found that with the BERT encoder, decreasing the learning rate by factor of 10 results in better reconstruction performance of VAE. We train the models with BERT encoder for 3 epochs.

Evaluation Metrics

Text Classification. To report the classification performance we use accuracy as a metric.

Sparsity. We measure Hoyer[11] (Hurley and Rickard, 2009) on the representations of all data points in a corpus and report its average as our sparsity metric (Mathieu et al., 2019). Hoyer, in a nutshell, is ratio of the L^2 to L^1 norm, normalised by the number of dimensions. Higher indicates more sparsity. More specifically, to evaluate the average Hoyer, or as we refer to it as Average Hoyer (AH) in the experiments, either on a validation or test corpus we employ the following procedure. First, for each x_i in the corpus $\{x_1,...,x_n\}$ we obtain its corresponding z_i by sampling it from a probabilistic encoder of a VAE model, such that for each x_i we sample one z_i: e.g. $x_1 \rightarrow z_1$. Then we normalise $\bar{z}_i = z_i/\sigma(z)$, where $z = \{z_1,...,z_n\}$, and $\sigma(.)$ is the standard deviation. Finally, for each $\bar{z}_i$ we compute Hoyer as follows:

$$Hoyer(\bar{z}_i) = \frac{\sqrt{d} - ||\bar{z}_i||_1/||\bar{z}_i||_2}{\sqrt{d}-1}, \tag{6.1}$$

where d is the dimensionality of $\bar{z}_i$. To report the Hoyer for the whole corpus we compute the Average Hoyer $= \frac{1}{N}\sum_i^N Hoyer(\bar{z}_i)$, where N is the number of data points in a test or validation corpus.

6.3.2 Text Classification

Prior to use of a VAE encoder in the classification experiment, we pretrained it using the full VAE model with the corresponding VAE's objective function on one of the target corpus: Yelp, Yahoo or DBpedia. We compare performance of the sparse latent representations with their dense counterparts on the three text classification tasks (Figure 6.2). The classifier that we use comprises of the two dense layer of 32D each with the Leaky ReLU (Maas, 2013) activation function. To establish whether the performance gain or loss on the tasks

[11] Note, the Hoyer metric does not give credit for actual zeros, only to distributions that are closer to sparse. In our work this is acceptable because spike distribution is Normal centered at zero, hence in practice we sample values that are very close to zero but not exactly zero.

is achieved thanks to the sparsity inductive bias, for all the VAE models and BERT we freeze the parameters of the encoder and only train the classifier which we put on top of the encoder. However, for the simple classifier model its text encoder is being trained together with the classifier. When the classifier, $p(y|x)$, is trained with a probabilistic VAE encoder we marginalise the latent variable(s). This is done for instance for HSVAE as,

$$p(y|x) = \int_{z,\gamma} p(y|z)q(z|x,\gamma)q_\phi(\gamma|x)dzd\gamma$$

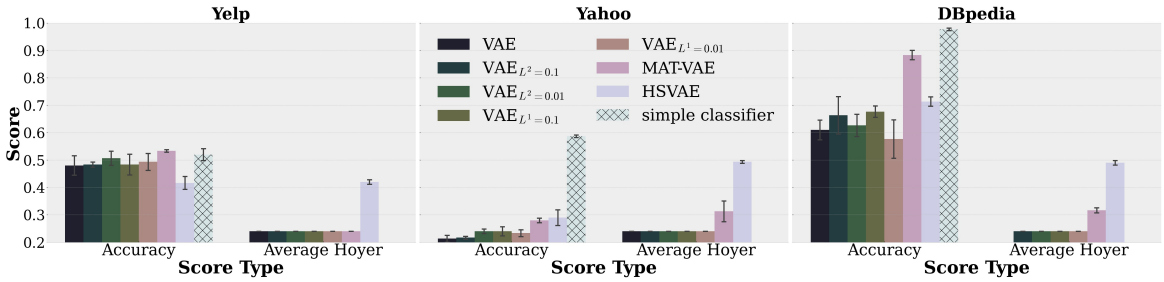
We approximate the integral with MC by taking $K = 5$ samples from the probabilistic encoder both to train and to test the classifier: For each x_i in a batch $\{x_1,...,x_p\}$:

1. sample K of $\gamma_{i,j}$ from $q_\phi(\gamma|x_i)$ i.e. a set of sampled γ's is $\{\gamma_{i,1},...,\gamma_{i,K}\}$
2. sample K of $z_{i,j}$ from $q_\phi(z|x_i,\gamma_{i,j})$ i.e. a set of sampled tuples of $z_{i,j}$ and $\gamma_{i,j}$ is $\{(z_{i,1},\gamma_{i,1}),...,(z_{i,K},\gamma_{i,K})\}$ in other words for each $\gamma_{i,j}$ we sample only one $z_{i,j}$.

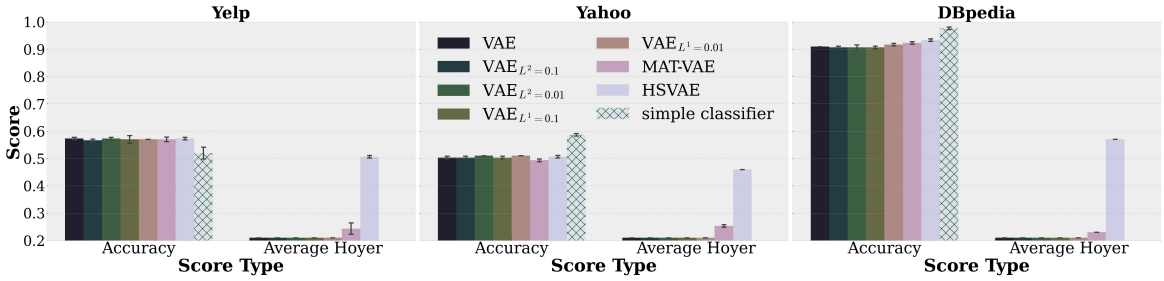
For the other VAEs the procedure is similar. With the MC approximation : $p(y|x) \approx 0.2 \times \sum_i^5 p(y|z_i)$.

For a systematic comparison of various VAEs, we collate classification performance of VAEs with comparable reconstruction loss - which indicates how informative the latent code is for the decoder during reconstruction. In other words the reconstruction loss serves as an intrinsic metric. Thus, for an example, in Figure 6.2a, for the Yelp corpus all the VAE models have a similar reconstruction loss. The same applies to Figure 6.2b and Figure 6.2c.

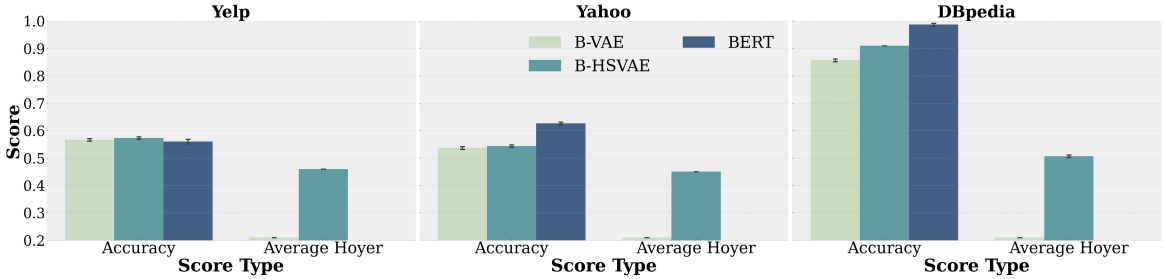
Comparing the accuracy of the classifiers that are trained with the different latent representations i.e. sparse and dense (Figure 6.2), shows that in general the performance of the sparse latent representations induced by HSVAE or MAT-VAE is on par with their dense latent counterparts inferred by the VAEs. However, the performance of HSVAE slightly lagging behind on the Yelp corpus when the dimensionality of the latent representation is 32D (Figure 6.2a). We put forward a hypothesis that may explain this in Subsection 6.3.4. Also, when the dimensionality of the latent representation is 32D, the accuracy of MAT-VAE is slightly better than of HSVAE, but this performance is reached at lower levels of sparsity.



(a)



(b)



(c)

Fig. 6.2 Classification Accuracy and Average Hoyer (higher means sparser z) for various VAE variants and the two baselines: simple classifier and BERT evaluated on Yelp, Yahoo or DBpedia test. The latent code of the VAEs is 32 D Figure (a) and 768 D Figures (b) and (c). Hoyer metric is not applicable to the simple classifier in the panels (a) and (b) and to the vanilla BERT model in the panel (c). The weights of the VAE encoders and BERT are **frozen** during the training of the classifiers. While the encoder of the simple classifier is updated during the training.

Additionally, we found that regularising the posterior parameters of the VAE model with either L^1 or L^2 norm, in some cases, helps to increase the classification accuracy, but does not reach AH higher than the vanilla VAE. Notably, the classification performance of all the VAE models becomes almost identical when the dimensionality of the latent space is increased from 32D to 768D, with HSVAE slightly outperforming all other VAEs on the DBpedia corpus (Figure 6.2b). We further elaborate on it in Subsection 6.3.4.

Use of BERT as an encoder, in our settings, only gives an improvement on the Yahoo corpus with B-HSVAE performing on par with B-VAE, but does not reach the classification accuracy of the plain BERT. We hypothesise that to reach the full potential of the use of a pretrained encoder in a VAE model one needs to pair it with a powerful decoder such as GPT-2 (Radford et al., 2019) as it is the case in the Li et al. (2020b) VAE model. Further exploration of this was beyond our compute resource.

Finally, one can observe that the simple classifier model performs on a par (in Figure 6.2a) or even worse (Figure 6.2b) than the VAE models on the Yelp corpus. Putting it into the context that the VAE encoders are not being trained with a supervision signal while the encoder of the simple classifier is, we speculate that this can be explained by the discussion put forward in Valpola (2014). A classifier in nature tries to remove all the information that is not relevant to the supervision signal, while an autoencoder tries to preserve as much as possible information in the latent code in order to reconstruct the original input data reliably. Thus, if the distribution of class related words in a text alone (see Subsection 6.3.4) is not indicative enough of a class then the classifier may perform poorly. In our case, we hypothesise that the VAE models capture some additional information other than class distribution of words in text that allows it to better discriminate the classes. For example, some class may have shorter sentences, on average, than the sentences presented in the other classes. This may provide an additional bias that allows the VAE models to discriminate sentences from this class from the sentences from the other classes. Thus, with this additional bias VAEs can perform better than the simple classifier. We leave this investigation for a future work.

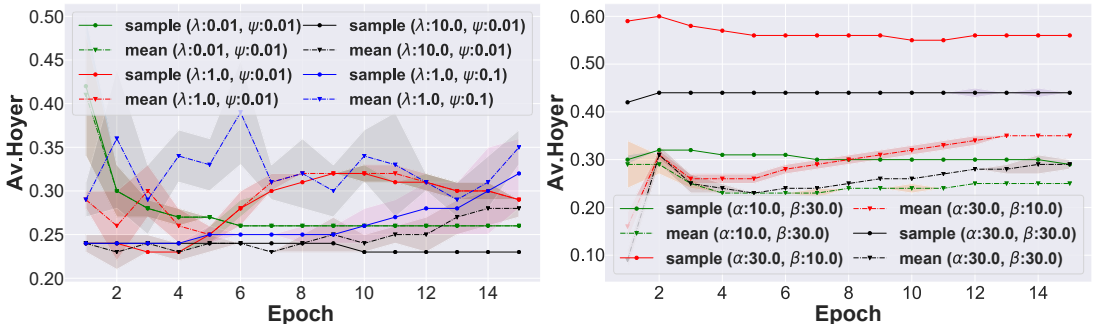


Fig. 6.3 Average Hoyer (Av.Hoyer; AH) on DBpedia corpus dev set for different parameterisations of Mathieu et al. (2019) (left) vs. HSVAE (right). Same is observed on Yelp and Yahoo (see *Appendix*). Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

6.3.3 Representation Sparsity

In Figure 6.3 we compare HSVAE with MAT-VAE. We report AH both on the mean and samples from the posterior distributions. As illustrated, MAT-VAE struggles to achieve steady and consistent AH regardless of the configurations of its hyperparameters (ψ, λ). However, HSVAE stably controls the level of sparsity with α and β parameters, a positive effect of its more flexible posterior distribution and the learnable distribution over γ.

6.3.4 Can Sparsity Patterns Encode Classes?

In order to identify pertinent features, the unsupervised representation learning models are typically trained/fine-tuned on corpora that are closely related to the downstream task. As such, without a supervisory signal, the model can only rely on the distribution of words in a text in order to identify these relevant features for the task. Ideally, compared to their dense counterparts, an unsupervised sparsification model such as HSVAE could result in performance improvement on downstream tasks if they capture the task-related features and discard the noisy features. However, if the sparsification model fail to capture the task related signal in its sparsity pattern; it can hurt the performance of the model on the downstream task as the task-related information can be removed. In what follows we investigate this direction

by analysing the sparsity patterns and relate this analysis to the classification performance of the model

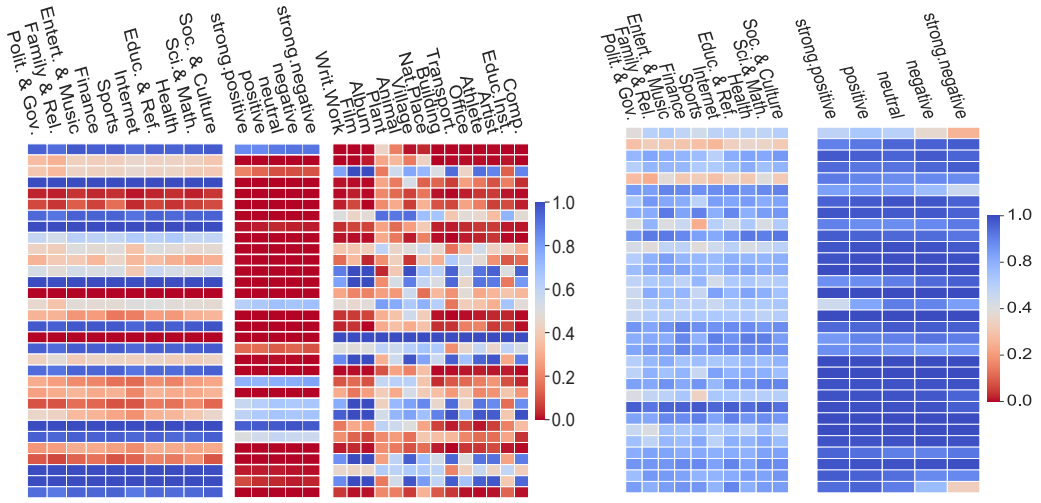
Analysis of γ. We hypothesise that if γ captures a class of a sentence then the sentences that belong to the same class should have a similar sparsity patterns in γ. To obtain a class specific γ_{class}, first, for each sentence x we obtain the mean of the posterior distribution: $q_\phi(\gamma|x)$ and we denote it as $\mu_{\gamma(x)}$. Then we binarise the mean such as $\mu^b_{\gamma(x)} = \text{Binarise}(\mu_{\gamma(x)})$, where Binarise$(\cdot)$ is defined as: 0 if $\mu_{\gamma(x)} < 0.5$ and 1 otherwise. Finally, for each class we average its $\mu^b_{\gamma(x)}$ vectors to obtain a single vector that represent this class: $\gamma_{class} = \frac{1}{M}\sum_{x \in class} \mu^b_{\gamma(x)}$, where M is a number of sentences in the class. The averaging removes the information that differentiate these sentences, while preserving the class information that is shared among them. A similar approach was also used in Mathieu et al. (2019).

Figure 6.4 reports the magnitudes of the γ_{class} vectors as heat maps for the three corpora. One would expect that γ_{class} of different classes should differ. For 32D γ_{class} (Figure 6.4a) this is the case when HSVAE is trained on the DBpedia and Yahoo but not on Yelp. Taking into account the unsupervised nature of these models, this difference is echoing the distribution of words in the classes, which is more distinct in DBpedia and Yahoo, but not in Yelp (see Subsection 6.3.4). We also hypothesis that this observation can explain inferior performance of the model on the Yelp corpus (Figure 6.2a).

In contrast, for γ_{class} in 768D (Figure 6.4b) one can observe that the different classes have different activation patterns even when HSVAE is trained on the Yelp corpus.[12] Also, the distributedness of the activation patterns now becomes more apparent when HSVAE is trained on the Yahoo corpus. This observation is also related to the distribution of words in the text (further elaborated in Subsection 6.3.4).

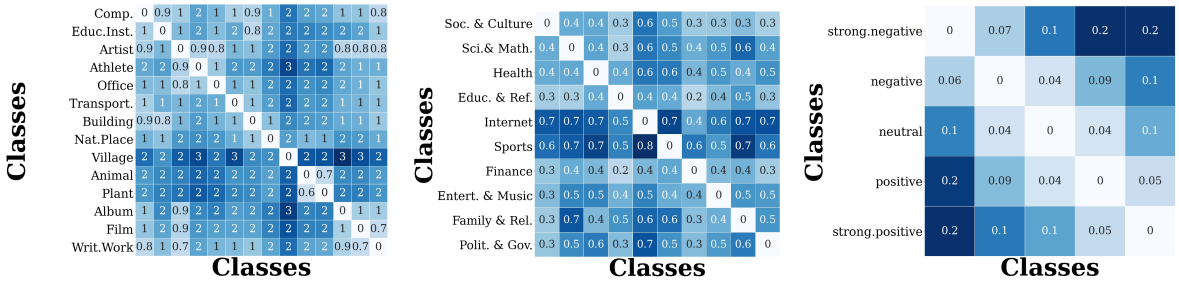
Intuitively, to reconstruct a sentence a VAE model first captures aspect of data that are the most conducive for reconstruction error reduction (Burgess et al., 2018). Therefore, given the limited dimensionality of the latent vector, the model will prioritised aspects of

[12]In Figure 6.4b we only show 32D out of 768D. This is one of the subsets of the 768 dimensions where the distributedness is present. It is not unique and the distributedness is also present in other dimensions of the 768D code.



(a) (b)

Fig. 6.4 Heat maps of γ_{class} (Subsection 6.3.4). (a) γ_{class} of 32D - from left to right: Yahoo, Yelp, DBpedia. (b) contiguous 32D out of 768D of γ_{class} - from left to right: Yahoo, Yelp.



(a) (b) (c)

Fig. 6.5 Experimental results for KL between classes on the three corpora: DBpedia (a), Yahoo (b) and Yelp (c).

data during encoding. As such, if the information such as sentence class is not strongly presented in the corpus the model could potentially ignore it during encoding. However, when the dimensionality of the latent space is increased, the model has more capacity to represent various aspects of data that may otherwise be ignored in the smaller dimensionality. We speculate this could explain the presence of distributedness of γ_{class} on Yelp for 768D as opposed to 32D, which also translates into matching the task performance of its dense counterpart (Figure 6.2b).

Class Kullback–Leibler Divergence

The question that has yet not been addressed is why in some cases the HSVAE model is more successful at capturing the class distribution when trained on DBpedia compared to Yelp. We previously hypothesised that the reason for this can be a word distribution in a text. To empirically test our hypothesis, we calculate the add-1 smoothed probabilities of words in the classes and measure the pairwise KL divergence across them. The magnitudes of the pairwise KL divergences are shown in Figure 6.5. As demonstrated, the magnitude of the KL divergence is the largest for DBpedia and smallest for Yelp. This indicates that separating classes in Yelp would rely on more subtle aspects of data, whereas surface-level cues are more present in DBpedia and allow for an easier discrimination.

6.4 Conclusion

When the sparsity inductive bias is employed in supervised learning settings the model can rely on a supervisory signal to identify features that are relevant to the task and remove the rest. However, in unsupervised settings - the setting which is now commonly used to pretrain the neural language models - the model can only rely on the distribution of words in a text in order to identify pertinent features. The absence of task, poses a series of reasonable questions. First, can we sparsify sentence embeddings with a sparsity inductive bias in unsupervised settings? Second, can sparse sentence embeddings, learned with sparsity inductive bias in an unsupervised setting, match (or outperform) the performance of their dense counterpart, and what are the necessary conditions for this to happen?

We studied these questions on three text classification tasks with a novel VAE model that we presented - Hierarchical Sparse Variational Autoencoder (HSVAE). HSVAE[13] uses

[13]Note, an alternative way to induce sparsity bias is to use discrete latent variables e.g. vectors of Bernoullis, or VQ-VAE (van den Oord et al., 2017). However, a disadvantage of the former approach is that the binary variables cannot model continuous information e.g. if a dimension of the latent vector encode a colour then the continuous dimension can be useful to further encode shade of the colour. For the later approach similar arguments can be put forward. Also, VQ-VAE does not sparsify the vectors in the dictionary that are inputted to the decoder. Studying how sparse vectors affect the decoding can be also interesting future work, which is possible with HSVAE.

sparsity inductive bias to learn a sparse latent representation of text. We demonstrated that using a prior distribution (Mathieu et al., 2019) that encourages sparsity does not necessarily transfer to inducing sparse representations for text, while HSVAE addresses this shortcoming by also using a more flexible posterior. Also, we showed that sparse representations, learned by HSVAE, are capable of encoding the underlying characteristics of a corpus (e.g. class labels), in their activation patterns. We established how statistical properties of a corpus such as a word distribution in a class and representation dimensionality of sentence embeddings affect the ability of learned sparse codes to represent task-related information. Finally, we showed how the presence/absence of task-related signal in the sparsity patterns affects the task performance of the model on the text classification tasks.

In this chapter, we only studied the effect the sparsity has on the discriminative tasks such as text classification. However, VAEs also allow us to perform generative tasks and can be used for text generation (see Chapter 5). Hence, in future work we plan to investigate if the sparsity patterns may allow us to use HSVAE for more controllable text generation (Hu et al., 2017). It was discussed in several works that sparsity can improve interpretability and disentanglement (Correia et al., 2019; Cui et al., 2019; Zhang et al., 2021). We expect, the generation to be more controllable thanks to, potentially, more interpretable sentence embeddings. Moreover, the objective of sparsity is to only keep the most important features relevant to the task and removing the rest - in other words removing the noise or spurious features. Hence, it potentially can be a factor that improves the performance on the out-of-distribution tasks; we plan to investigate if the sparsity can help us on out-of-distribution tasks (McCoy et al., 2020b).

In the next chapter we provide a summary of the contributions of this thesis and future research directions and questions emerging from this work.

7

Conclusion and Future Directions

Contemporary neural language models that are trained on a large amount of data achieve SOTA performance on many downstream tasks that require understanding of the meaning of language units. Moreover, they have been shown to acquire certain knowledge of syntax and semantics.

Despite the achieved success these models still have a lot of drawbacks (see Chapter 2). One way to advance their linguistic competence is to train them on even bigger corpora. This approach is attractive because it does not require any expertise in the design of the models. However, if a model requires exponentially more data to acquire certain knowledge/skill (e.g. reasoning) to become competent in a natural language understanding task (Warstadt et al., 2020; Zhang et al., 2020b) then it becomes impractical to train such a model.

Another alternative which we explore in this thesis is the use of inductive biases. Inductive biases allow a learning algorithm to prefer one solution over the alternatives. Hence, we may need less data to train a neural network as we explicitly state preferable solutions. Also, with inductive biases we can incorporate desirable properties that we may want a neural network to have e.g. sparse activation patterns.

7.1 Summary

We presented four lines of work that address various existing limitations in the learning of word and sentence embeddings.

7.1.1 RQ 1: Relational Inductive Bias for Words (Data-Based)

One such limitation that we discussed concerns learning a reliable word embedding for infrequent or unseen words in a corpus. To assign a meaning to a word the neural model needs to select a meaning out of many possible alternatives (not to mention homonymy/polysemy), and without a bias that would allow the model to narrow down the alternatives, it would rely on the sufficient occurrences of the word to create an adequate representation of word meaning. In Chapter 3, we proposed to use a KG-based relational inductive bias to learn embeddings of rare and unseen words. Our approach used a graph embedding technique that allows us to derive a semantic representation of a word in terms of relationships that exist between the word and other words in a KG. Also, we used a cross-lingual vector space transformation technique in order to merge lexical knowledge encoded in KGs with that derived from corpus statistics. We showed the reliability of our approach by evaluating the induced embeddings on multiple word similarity benchmarks as well as on a downstream NLP evaluation framework.

7.1.2 RQ 2: Relational Inductive Bias for Sentences (Data-Based)

In the remaining works, we switched focus from word embeddings to sentence embeddings. Scaling the distributed representations to larger language units such as phrases and sentences is still a non-trivial task (see Subsection 2.2.2). One reason for this is that there is still no known task that would allow us to effectively learn the composition of words in sentences. Without such a task there are numerous possible meanings that can be expressed by composing the words in sentences/phrases. One way to approach this challenge is via supervised learning where the supervisory signal expresses (or biases) the meaning of the sentences. Inspired by Hill et al. (2015a), who proposed to relate the meaning of word embeddings with embeddings of phrases via lexical resources, in Chapter 4 we presented a novel method for mapping text to KG entities by framing the task as a sequence-to-sequence problem. Instead of relying on the semantic representation of word embeddings as in Hill et al. (2015a), we used the structure of KG that at the same time allows us to reduce possible meanings a

phrase/sentence can have and also allows us to ground the meaning of the phrase/sentences. The encouraging results, comparable to those of state-of-the-art systems, is an indicator that our sentence embeddings do incorporate the semantics of the path graph.

Another gap in the induction of sentence embeddings is that we poorly understand what properties the sentence embeddings need to have in order to represent a sentence. In Chapters 5 and 6 we studied and controlled two properties of sentence embeddings: the amount of information they contain and sparsity. To impose these properties we incorporated the corresponding inductive biases via the VAE framework.

7.1.3 RQ 3: Information-Theoretic Inductive Bias for Sentences (Data-Agnostic)

Autoencoders are popular for unsupervised representation learning. In principle, autoencoders try to preserve as much as possible information about the data they model. However, it is yet poorly understood how much information should be preserved. In Chapter 5, we explored a VAE model from information-theoretic perspective. This perspective allowed us to treat the terms of the ELBO function as bounds on the mutual information between a sentence and its embedding. By controlling the bounds, we regulated the amount of information the learned representation stores about a sentence. We analysed the effect of this bias on the quality of the learned sentence representation via two downstream tasks: text generation and text classification.

7.1.4 RQ 4: Sparsity Inductive Bias for Sentences (Data-Agnostic)

As we discussed in Subsection 2.3.6 sparsity can allow us to reflect many linguistic properties in the neural language models and embeddings. However, can sparse sentence embeddings, learned with sparsity inductive bias, match (or outperform) the performance of their dense counterpart on downstream tasks? What are the necessary conditions for this to happen? In Chapter 6, we proposed a novel Hierarchical Sparse Variational Autoencoder, that imposes sparsity on sentence representations via direct optimisation of ELBO. We looked at the

implications of sparsity on text classification across three datasets and highlighted a link between the performance of sparse latent representations on downstream tasks and its ability to encode task-related information. Our frameworks, while achieving sparsity, allowed efficient utilisation of the representation space without compromising task performance.

In this thesis, we initiate the discussion of the proposed inductive biases. Moving forward, we outline possible directions in the next section.

7.2 Future Directions

Current SOTA representation learning neural models that do not use inductive biases do not perform well on commonsense reasoning/language understanding tasks (Zhang et al., 2019; Ding et al., 2020; Yamada et al., 2020). Zhang et al. (2020b) explain this shortcoming based on the large amount of unstructured text that is needed to acquire such fine-grained knowledge. KG, in turn, can encode commonsense knowledge about the world, in a structured form, and could potentially help neural networks in the aforementioned tasks. However, how to effectively bias the models with the structural information encoded in the KG is still an open question. Most of the existing works (Zhang et al., 2019; Kalinowski and An, 2020) use relational knowledge graph embedding techniques such as TransE (Bordes et al., 2013) where the emphasis is put on modeling the relationships between the entities in a KG in a form of triplets, while utilising the structure of KGs is not very well explored. One can investigate how the proposed approach in Chapter 4 can benefit sequence-to-sequence models such as T5 when used as an auxiliary task to train the model - in other words whether this additional task improves the transfer learning.[1]

Given a language unit such as sentence one can observe (performing either syntactic or semantic analysis of the sentence) that the interaction (both syntactic and semantic) between

[1] In Chapter 4 we proposed to use a KG structure as a supervision signal to learn the meaning of sentences. The supervision signal is represented as path graph that encodes topological dependencies that exist between entities of the KG. Such a relation between two sequences: text and path graph can me modeled by a sequence-to-sequence model. Moreover, we showed that the sequence-to-squence models do encode the structural information of the KG in the sentence embeddings. The formulation of the task and our empirical results demonstrates how the sequence-to-sequence models such as T5 could be useful to extract the commonsense knowledge from structure resource such as KG.

its smaller units - words - is sparse. Thus sparsity inductive bias can allow us to capture this property in modelling phase. Guiding the representation learning to preserve the syntactic and semantic connections and remove the spurious ones, is a potential area of future work. Also, from a practical (downstream task oriented) point of view, removing the spurious information can potentially benefit the generalisation of the model on out-of-distribution tasks (McCoy et al., 2020b), where currently SOTA models struggle. A potential approach could be the incorporation of explicit linguistic biases into the learned representations with the group sparsity (Huang and Zhang, 2010). Yogatama et al. (2015) take inspiration from hierarchical organisation of words/concepts in the brain and propose to group activation of the dimensions of word embeddings according to a tree structure. Future works can take inspiration from the group sparsity that has been done for the other domains (Cevher et al., 2009; Andersen et al., 2014). For example, Andersen et al. (2014) propose to use Gaussian process (Rasmussen and Williams, 2005) together with the Spike-and-Slab distribution to structure the sparsity patterns for modelling of electroencephalogram (EEG) data. To adapt this work to text, one can take an inspiration form Gaussian processes for text (Beck, 2017).

Moreover, sparsity can potentially allow us to interpret[2] the internal representations of the models e.g., attention (Bahdanau et al., 2014) or word/sentence embeddings more easily. To what extend this is indeed the case has been debated by the community. Some question the role of sparsity in interpretability (Kim et al., 2014; Meister et al., 2021) while others find it to be a useful proxy to interpretability (Correia et al., 2019; Cui et al., 2019; Treviso and Martins, 2020). Indeed, in our work (see Chapter 6) sparsity patterns learned by the model helped us understand when sparse sentence embeddings fail to perform on par with dense sentence embeddings.

However, there is still an obstacle that can make it harder for one to interpret the sparse sentence embeddings - the dimensions that are active can be entangled and thus it can be difficult to interpret how each active dimension influences the output of the model. Two approaches that can potentially address this issue are disentanglement (Higgins et al., 2018)

[2]By interpretability, here, we mean being able to analyse a relation between internal representations of the model and output that the model produces. For example, how a change in a value of a certain dimension of a sentence embedding results in the change of the output that the model produces.

and group sparsity - activate dimensions in groups, where we can interpret why each group of dimensions has been activated (Yogatama et al., 2015).

In future work, we also plan to further explore the interpretability aspect of the sentence embedding learned by HSVAE model. A task the we plan to use is controllable text generation (Hu et al., 2017). Potentially by changing the sparsity patterns and/or experimenting with the values of active dimensions we can generate sentences with certain properties. For example, if we can find that a certain sparsity pattern is responsible for the tense of a sentence we can use this pattern to generate the sentence in the intended tense.

Finally, adaptive sparsity has been shown to be beneficial for unsupervised disentanglement in text (Zhang et al., 2021). This is encouraging because the aforementioned interpretability issue of active dimensions of sparse representations may be not so severe. A potential reason for this is that sparsity allows to utilise different sub-regions of the representation space which in turn may lead to decoupling (disentanglement) of the dimensions. However, the scope of experiments conducted by Zhang et al. (2021) that regards sparse sentence embeddings is limited. Authors use only one model (Mathieu et al., 2019) that biases sentence embeddings towards sparsity. In further studies, we plan to conduct a more thorough investigation of this phenomenon by employing more models with sparsity inductive bias. For example, similar experiments conducted by Locatello et al. (2019) would allow us to better understand the role of sparsity in disentanglement.

References

Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A Benchmarking Platform for Text Generation Models. In *SIGIR*.

C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

Z. Harris. Distributional structure. volume 10, pages 146–162, 1954.

N. Goodman. The new riddle of induction. *Fact, Fiction and Forecast*, pages 59–83, 1955.

J. Firth. A Synopsis of Linguistic Theory 1930-1955. In *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957.

N. Chomsky. *Aspects of the Theory of Syntax*. The MIT Press, 1965.

H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633, 1965.

A. M. Collins and M. R. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8:240–247, 1969.

R. Montague. The proper treatment of quantification in ordinary english. In P. Suppes, J. Moravcsik, and J. Hintikka, editors, *Approaches to Natural Language*, pages 221–242. Dordrecht, 1973.

D. Hume. *Treatise of human nature*. Oxford University Press, 1978.

T. M. Mitchell. The Need for Biases in Learning Generalizations. Technical report, 1980.

D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

T. J. Mitchell and J. J. Beauchamp. Bayesian Variable Selection in Linear Regression. *Journal of the American Statistical Association*, pages 1023–1032, 1988.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, 1988.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1: 541–551, 1989.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179 – 211, 1990.

References

J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46:77–105, 1990.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

R. Caruana. Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Morgan Kaufmann, 1993.

R. Davis, H. E. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14:17–33, 1993.

D. Gordon and M. desJardins. Evaluation and selection of biases in machine learning. *Machine Learning*, 20:5–22, 1995.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of ICML*, pages 331–339, 1995.

C. Goller and A. Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 347–352, 1996.

B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9, 1996.

D. H. Wolpert. The Lack of a Priori Distinctions between Learning Algorithms. *Neural Comput.*, 8:1341–1390, 1996.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9:1735–1780, 1997.

B. M. Janssen. Compositionality with an appendix by b. 1997.

D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *Trans. Evol. Comp*, 1:67–82, 1997.

C. Fellbaum, editor. *WordNet: An Electronic Database*. MIT Press, 1998.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

D. Barber and F. Agakov. The IM algorithm: a variational approach to Information Maximization. In *NIPS 2003*, 2003.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *JMLR*, 3, 2003.

B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of ACL*, pages 51–61, 2004.

P. Blackburn and J. Bos. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Center for the Study of Language and Information, 2005.

H. Ishwaran and J. S. Rao. Spike and slab variable selection: Frequentist and Bayesian strategies. *The Annals of Statistics*, 2005.

B. Pang and L. Lee. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of ACL*, pages 115–124, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

S. Akaho. A kernel method for canonical correlation analysis. *CoRR*, abs/cs/0609071, 2006.

A. Alexandrescu and K. Kirchhoff. Factored Neural Language Models. In *Proceedings of HLT-NAACL*, pages 1–4, 2006.

D. Greene and P. Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of ICML*, pages 377–384. ACM, 2006.

M. Creutz and K. Lagus. Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Trans. on Speech and Language Processing*, 4:3:1–3:34, 2007.

H. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22: 400–407, 2007.

V. Nastase. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 763–772, 2008.

P. N. Robinson, S. Köhler, S. B. Bauer, D. Seelow, D. Horn, and S. Mundlos. The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *American journal of human genetics*, 83:610–5, 2008.

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of HLT-NAACL*, pages 19–27, 2009.

Y. Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2:1–127, 2009.

V. Cevher, M. Duarte, C. Hegde, and R. Baraniuk. Sparse Signal Recovery Using Markov Random Fields. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

References

N. Hurley and S. Rickard. Comparing Measures of Sparsity. *IEEE Transactions on Information Theory*, 55:4723–4741, 2009.

F. Scarselli, M. Gori, A. Tsoi, M. Hagenbuchner, and G. Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

T. Griffiths, N. Chater, C. Kemp, A. Perfors, and J. Tenenbaum. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14:357–364, 2010.

J. Huang and T. Zhang. The Benefit of Group Sparsity. *The Annals of Statistics*, pages 1978–2004, 2010. URL http://www.jstor.org/stable/20744481.

T. Jaeger. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology*, 61:23–62, 2010.

J. Reisinger and R. J. Mooney. Multi-Prototype Vector-Space Models of Word Meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

C. Shaoul and C. Westbury. The Westbury Lab Wikipedia Corpus. 2010. Accessed: 2016-11-10.

R. Socher, C. D. Manning, and A. Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *In Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 2010.

J. Turian, L.-A. Ratinov, and Y. Bengio. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

D. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, High Performance Convolutional Neural Networks for Image Classification. pages 1237–1242, 2011.

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning Word Vectors for Sentiment Analysis. In *Proceedings of ACL-HLT*, pages 142–150, 2011.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161. Association for Computational Linguistics, 2011.

I. Sutskever, J. Martens, and G. Hinton. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

W. Blacoe and M. Lapata. A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.

T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren. Large-scale Learning of Word Relatedness with Constraints. In *Proceedings of KDD*, pages 1406–1414, 2012.

A. Raposo, M. Mendes, and J. F. Marques. The hierarchical organization of semantic memory: Executive function in the processing of superordinate concepts. *NeuroImage*, 59: 1870–1878, 2012.

L. M. Schriml, C. Arze, S. Nadendla, Y.-W. W. Chang, M. Mazaitis, V. Felix, G. Feng, and W. A. Kibbe. Disease Ontology: a backbone for disease semantic integration. In *Nucleic Acids Research*, 2012.

X. Zhang, J. Zhou, C. Wang, C. Li, and L. Song. Multi-class support vector machine optimized by inter-cluster distance and self-adaptive deferential evolution. *Applied Mathematics and Computation*, 218:4973–4987, 2012.

G. Andrew, R. Arora, K. Livescu, and J. Bilmes. Deep Canonical Correlation Analysis. In *Proceedings of ICML*, 2013.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Association for Computational Linguistics, 2013.

Y. Bengio, A. C. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35:1798–1828, 2013.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

A. Graves. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 2013.

A. Lazaridou, M. Marelli, R. Zamparelli, and M. Baroni. Compositionally Derived Representations of Morphologically Complex Words in Distributional Semantics. In *Proceedings of ACL*, pages 1517–1526, 2013.

T. Luong, R. Socher, and C. Manning. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of CoNLL*, pages 104–113, 2013.

A. L. Maas. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 2013.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Workshop at ICLR*, 2013.

References

M. T. Pilehvar, D. Jurgens, and R. Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1351, 2013.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. Parsing With Compositional Vector Grammars. In *Proceedings of EMNLP*, pages 455–465, 2013.

M. Andersen, O. Winther, and L. Hansen. Bayesian inference for structured spike and slab priors. In *Proceedings of the 28th Annual Conference on Advances in Neural Information Processing Systems 27*, pages 1745–1753. Neural Information Processing Systems Foundation, 2014.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

J. A. Botha and P. Blunsom. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of ICML*, pages 1899–1907, 2014.

E. Bruni, N. K. Tran, and M. Baroni. Multimodal Distributional Semantics. *JAIR*, 49:1–47, 2014.

K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics, 2014a.

K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014b.

C. N. Dos Santos and B. Zadrozny. Learning Character-level Representations for Part-of-speech Tagging. In *Proceedings of ICML*, pages II–1818–II–1826, 2014.

M. Faruqui and C. Dyer. Improving Vector Space Word Representations Using Multilingual Correlation. In *Proceedings of EACL*, pages 462–471, 2014.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A Convolutional Neural Network for Modelling Sentences. *CoRR*, abs/1404.2188, 2014. URL http://arxiv.org/abs/1404.2188.

B. Kim, C. Rudin, and J. Shah. The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In *NIPS*, 2014.

Y. Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

A. Moro, A. Raganato, and R. Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2: 231–244, 2014.

A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069. Association for Computational Linguistics, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. page 3104–3112, 2014.

H. Valpola. From neural PCA to deep unsupervised learning. *CoRR*, abs/1411.7783, 2014.

M. Ballesteros, C. Dyer, and N. A. Smith. Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs. In *Proceedings of EMNLP*, pages 349–359, 2015.

S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015a.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *CoRR*, abs/1511.06349, 2015b. URL http://arxiv.org/abs/1511.06349.

Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *CoRR*, abs/1509.00519, 2015.

S. Cao, W. Lu, and Q. Xu. GraRep: Learning Graph Representations with Global Structural Information. In *Proceedings of CIKM*, pages 891–900, 2015.

M. Faruqui and C. Dyer. Non-distributional Word Vector Representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 464–469. Association for Computational Linguistics, 2015.

M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith. Sparse Overcomplete Word Vector Representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500. Association for Computational Linguistics, 2015.

F. Hill, K. Cho, A. Korhonen, and Y. Bengio. Learning to Understand Phrases by Embedding the Dictionary. *CoRR*, abs/1504.00548, 2015a. URL http://arxiv.org/abs/1504.00548.

F. Hill, R. Reichart, and A. Korhonen. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, 41:665–695, 2015b.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2015.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-Thought Vectors. *CoRR*, abs/1506.06726, 2015.

References

W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, and T. Luis. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of EMNLP*, pages 1520–1530, 2015.

M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs: From Multi-Relational Link Prediction to Automated Knowledge Graph Construction. *CoRR*, abs/1503.00759, 2015.

M. T. Pilehvar and R. Navigli. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128, 2015.

I. Sergienya and H. Schütze. Learning Better Embeddings for Rare Words Using Distributional Representations. In *Proceedings of EMNLP*, pages 280–285, 2015.

R. Soricut and F. Och. Unsupervised Morphology Induction Using Word Embeddings. In *Proceedings of NAACL-HLT*, pages 1627–1637, 2015.

K. S. Tai, R. Socher, and C. D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics, 2015.

W.-t. Yih, M.-W. Chang, X. He, and J. Gao. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331, 2015.

D. Yogatama, M. Faruqui, C. Dyer, and N. A. Smith. Learning Word Representations with Hierarchical Sparse Coding. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37, pages 87–96. JMLR.org, 2015.

X. Zhang, J. Zhao, and Y. LeCun. Character-Level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657. MIT Press, 2015.

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to Compose Neural Networks for Question Answering. *CoRR*, abs/1601.01705, 2016.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational Lossy Autoencoder. *CoRR*, abs/1611.02731, 2016.

T. Dozat. Incorporating Nesterov Momentum into Adam. 2016.

C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics, 2016.

L. Ehrlinger and W. Wöß. Towards a definition of knowledge graphs. In *SEMANTiCS*, 2016.

D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of EMNLP*, pages 2173–2182, 2016.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

A. Grover and J. Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of KDD*, pages 855–864, 2016.

F. Hill, K. Cho, and A. Korhonen. Learning Distributed Representations of Sentences from Unlabelled Data. *CoRR*, abs/1602.03483, 2016.

M. H. . M. Johnson. Elbo surgery. *Workshop on Advances in Approx*, 2016. URL http://approximateinference.org/accepted/HoffmanJohnson2016.pdf.

D. P. Kingma, T. Salimans, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *CoRR*, abs/1606.04934, 2016.

C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *International Conference on Learning Representations, ICLR*, 2016.

M. Nickel, L. Rosasco, and T. Poggio. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961, 2016.

M. T. Pilehvar and N. Collier. Improved Semantic Representation for Domain-Specific Entities. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 12–16. Association for Computational Linguistics, 2016.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics, 2016.

R. Sennrich, B. Haddow, and A. Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. Association for Computational Linguistics, 2016a.

R. Sennrich, B. Haddow, and A. Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics, 2016b.

F. Sun, J. Guo, Y. Lan, J. Xu, and X. Cheng. Sparse Word Embeddings Using l1 Regularized Online Learning. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2915–2921. IJCAI/AAAI Press, 2016.

S. Upadhyay, M. Faruqui, C. Dyer, and D. Roth. Cross-lingual Models of Word Embeddings: An Empirical Comparison. In *Proceedings of ACL*, pages 1661–1670, 2016.

References

D. Wang, P. Cui, and W. Zhu. Structural Deep Network Embedding. In *Proceedings of KDD*, pages 1225–1234, 2016.

J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *ICLR*, 2016.

O. Abend and A. Rappoport. The State of the Art in Semantic Representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89. Association for Computational Linguistics, 2017.

S. Arora, Y. Liang, and T. Ma. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *5th International Conference on Learning Representations*, 2017.

D. Bahdanau, T. Bosc, S. Jastrzebski, E. Grefenstette, P. Vincent, and Y. Bengio. Learning to Compute Word Embeddings On the Fly. *CoRR*, abs/1706.00286, 2017.

D. E. Beck. Gaussian Processes for Text Regression. 2017.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

H. Cai, V. W. Zheng, and K. C. Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications. *CoRR*, abs/1709.07604, 2017.

H. Chen, S. Huang, D. Chiang, and J. Chen. Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945. Association for Computational Linguistics, 2017.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *CoRR*, abs/1705.02364, 2017.

T. Dozat and C. D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*, 2017.

Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward Controlled Generation of Text. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2017.

M. J. Kusner, B. Paige, and J. Miguel HernÃ¡ndez-Lobato. Grammar Variational Autoencoder. 2017.

R. Jia and P. Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics, 2017.

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

A. Lazaridou, M. Marelli, and M. Baroni. Multimodal Word Meaning Induction From Minimal Exposure to Natural Text. *Cognitive Science*, 41:677–705, 2017.

B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in Translation: Contextualized Word Vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

M. T. Pilehvar and N. Collier. Inducing Embeddings for Rare and Unseen Words by Leveraging Lexical Resources. In *Proceedings of the EACL*, pages 388–393, 2017.

R. Speer, J. Chin, and C. Havasi. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. pages 4444–4451, 2017.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural Discrete Representation Learning. In *NIPS*, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29:2724–2743, 2017.

D. Weissenborn. Reading Twice for Natural Language Understanding. *CoRR*, abs/1706.02596, 2017.

Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved Variational Autoencoders for Text Modeling Using Dilated Convolutions. page 3881–3890, 2017.

S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei. Tackling Over-pruning in Variational Autoencoders. *International Conference on Machine Learning: Workshop on Principled Approaches to Deep Learning*, 2017.

S. Zhao, J. Song, and S. Ermon. InfoVAE: Information Maximizing Variational Autoencoders. *CoRR*, abs/1706.02262, 2017.

A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy. Fixing a Broken ELBO. In *ICML*, 2018.

S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear Algebraic Structure of Word Senses, with Applications to Polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

G. Barello, A. S. Charles, and J. W. Pillow. Sparse-Coding Variational Auto-Encoders. *bioRxiv*, 2018.

References

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

Y. Belinkov and J. R. Glass. Analysis Methods in Neural Language Processing: A Survey. *CoRR*, abs/1812.08951, 2018.

L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60:223–311, 2018.

C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-VAE. *CoRR*, abs/1804.03599, 2018.

J. Camacho-Collados and M. T. Pilehvar. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal of Artificial Intelligence Research*, 2018.

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174. Association for Computational Linguistics, 2018.

O. Cífka, A. Severyn, E. Alfonseca, and K. Filippova. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *CoRR*, abs/1804.07972, 2018.

A. Fan, M. Lewis, and Y. N. Dauphin. Hierarchical Neural Story Generation. *CoRR*, abs/1805.04833, 2018.

M. Figurnov, S. Mohamed, and A. Mnih. Implicit Reparameterization Gradients. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 439–450. Curran Associates Inc., 2018.

I. Higgins, D. Amos, D. Pfau, S. Racanière, L. Matthey, D. J. Rezende, and A. Lerchner. Towards a Definition of Disentangled Representations. *CoRR*, abs/1812.02230, 2018.

D. Kartsaklis, M. T. Pilehvar, and N. Collier. Mapping Text to Knowledge Graph Entities using Multi-Sense LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1959–1970. Association for Computational Linguistics, 2018.

Y. Mao, X. Ren, J. Shen, X. Gu, and J. Han. End-to-End Reinforcement Learning for Automatic Taxonomy Induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2462–2472. Association for Computational Linguistics, 2018.

R. Marvin and T. Linzen. Targeted Syntactic Evaluation of Language Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics, 2018.

P. Minervini, M. Bosnjak, T. Rocktäschel, and S. Riedel. Towards Neural Theorem Proving at Scale. *CoRR*, abs/1807.08204, 2018.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

M. T. Pilehvar, D. Kartsaklis, V. Prokhorov, and N. Collier. CARD-660: Cambridge Rare Word Dataset - a Reliable Benchmark for Infrequent Word Representation Models. In *Proceedings of EMNLP*, 2018.

B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker. On variational lower bounds of mutual information. In *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

S. Ruder. A Review of the Neural History of Natural Language Processing. 2018.

A. Subramanian, D. Pruthi, H. Jhamtani, T. Berg-Kirkpatrick, and E. H. Hovy. SPINE: SParse Interpretable Neural Embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4921–4928, 2018a.

S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. *CoRR*, abs/1804.00079, 2018b.

J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. pages 809–819, 2018.

J. M. Tomczak and M. Welling. VAE with a VampPrior. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 1214–1223*, 2018.

K. Tran, A. Bisazza, and C. Monz. The Importance of Being Recurrent for Modeling Hierarchical Structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736. Association for Computational Linguistics, 2018.

V. Trifonov, O. Ganea, A. Potapenko, and T. Hofmann. Learning and Evaluating Sparse Interpretable Sentence Embeddings. In T. Linzen, G. Chrupala, and A. Alishahi, editors, *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP*, pages 200–210. Association for Computational Linguistics, 2018.

F. Wagner, Y. Yan, and I. Yanai. K-nearest neighbor smoothing for high-throughput single-cell RNA-Seq data. *bioRxiv*, 2018.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355. Association for Computational Linguistics, 2018.

K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 1039–1050. Curran Associates Inc., 2018.

References

N. Zaslavsky, C. Kemp, T. Regier, and N. Tishby. Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences*, 115:7937–7942, 2018.

Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, and W. Dolan. Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization. In *NeurIPS*, 2018.

A. Alishahi, G. Chrupala, and T. Linzen. Analyzing and Interpreting Neural Networks for NLP: A Report on the First BlackboxNLP Workshop. *CoRR*, abs/1904.04063, 2019. URL http://arxiv.org/abs/1904.04063.

D. Bahdanau, S. Murty, M. Noukhovitch, T. H. Nguyen, H. de Vries, and A. Courville. Systematic Generalization: What Is Required and Can It Be Learned? In *International Conference on Learning Representations*, 2019.

M. Baroni. Linguistic generalization and compositionality in modern artificial neural networks. *CoRR*, abs/1904.00157, 2019. URL http://arxiv.org/abs/1904.00157.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating Long Sequences with Sparse Transformers. *CoRR*, abs/1904.10509, 2019.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What Does BERT Look at? An Analysis of BERT's Attention. In *BlackboxNLP@ACL*, 2019.

A. Coenen, E. Reif, A. Yuan, B. Kim, A. Pearce, F. Viégas, and M. Wattenberg. Visualizing and Measuring the Geometry of BERT. In *NeurIPS*, 2019.

G. M. Correia, V. Niculae, and A. F. T. Martins. Adaptively Sparse Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184. Association for Computational Linguistics, 2019.

R. Cotterell, C. Kirov, M. Hulden, and J. Eisner. On the Complexity and Typology of Inflectional Morphological Systems. *Transactions of the Association for Computational Linguistics*, 7:327–342, 2019.

B. Cui, Y. Li, M. Chen, and Z. Zhang. Fine-tune BERT with Sparse Self-Attention Mechanism. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3548–3553. Association for Computational Linguistics, 2019.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

R. Futrell, P. Qian, E. Gibson, E. Fedorenko, and I. Blank. Syntactic dependencies correspond to word pairs with high mutual information. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 3–13. Association for Computational Linguistics, 2019.

J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *ICLR*, 2019.

R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

A. Holtzman, J. Buys, M. Forbes, and Y. Choi. The Curious Case of Neural Text Degeneration. *CoRR*, abs/1904.09751, 2019.

Y. Kim, A. Rush, L. Yu, A. Kuncoro, C. Dyer, and G. Melis. Unsupervised Recurrent Neural Network Grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117. Association for Computational Linguistics, 2019.

A. Kuncoro, C. Dyer, L. Rimell, S. Clark, and P. Blunsom. Scalable Syntax-Aware Language Models Using Knowledge Distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3472–3484. Association for Computational Linguistics, 2019.

W. Li and S. Hao. Sparse Lifting of Dense Vectors: Unifying Word and Sentence Representations. *CoRR*, abs/1911.01625, 2019.

P. Liu, S. Chang, X. Huang, J. Tang, and J. C. K. Cheung. Contextualized Non-Local Neural Networks for Sequence Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6762–6769, 2019a.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692, 2019b.

F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. 97:4114–4124, 2019.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling Disentanglement in Variational Autoencoders. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 4402–4412. PMLR, 2019.

T. McCoy, E. Pavlick, and T. Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448. Association for Computational Linguistics, 2019.

T. Niven and H.-Y. Kao. Probing Neural Network Comprehension of Natural Language Arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664. Association for Computational Linguistics, 2019.

References

A. Ormazabal, M. Artetxe, G. Labaka, A. Soroa, and E. Agirre. Analyzing the Limitations of Cross-lingual Word Embedding Mappings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4990–4995. Association for Computational Linguistics, 2019.

T. Pelsmaeker and W. Aziz. Effective Estimation of Deep Generative Language Models. *CoRR*, abs/1904.08194, 2019.

V. Prokhorov, E. Shareghi, Y. Li, M. T. Pilehvar, and N. Collier. On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 118–127. Association for Computational Linguistics, 2019.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

S. Ravfogel, Y. Goldberg, and T. Linzen. Studying the Inductive Biases of RNNs with Synthetic Variations of Natural Languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3532–3542. Association for Computational Linguistics, 2019.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

Y. Shen, S. Tan, A. Sordoni, and A. Courville. Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks. In *International Conference on Learning Representations*, 2019.

F. Tonolini, B. S. Jensen, and R. Murray-Smith. Variational Sparse Coding. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019.

A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

P. Xu, J. C. K. Cheung, and Y. Cao. On Variational Learning of Controllable Representations for Text without Supervision. *arXiv preprint arXiv:1905.11975*, 2019.

Z. Ye, Q. Guo, Q. Gan, X. Qiu, and Z. Zhang. BP-Transformer: Modelling Long-Range Context via Binary Partitioning. *CoRR*, abs/1911.04070, 2019.

D. Yogatama, C. de Masson d'Autume, J. Connor, T. Kocisky, M. Chrzanowski, L. Kong, A. Lazaridou, W. Ling, L. Yu, C. Dyer, and P. Blunsom. Learning and Evaluating General Linguistic Intelligence. *CoRR*, abs/1901.11373, 2019.

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451. Association for Computational Linguistics, 2019.

G. Zhao, J. Lin, Z. Zhang, X. Ren, Q. Su, and X. Sun. Explicit Sparse Transformer: Concentrated Attention Through Explicit Selection. *CoRR*, abs/1912.11637, 2019.

J. Andreas. Good-Enough Compositional Data Augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566. Association for Computational Linguistics, 2020.

V. Balasubramanian, I. Kobyzev, H. Bahuleyan, I. Shapiro, and O. Vechtomova. Polarized-VAE: Proximity Based Disentangled Representation Learning for Text Generation. *CoRR*, abs/2004.10809, 2020.

E. M. Bender and A. Koller. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198. Association for Computational Linguistics, 2020.

D. Ding, F. Hill, A. Santoro, and M. Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *CoRR*, abs/2012.08508, 2020.

H. Gholamalinezhad and H. Khosravi. Pooling Methods in Deep Neural Networks, a Review. 2020.

R. Girdhar and D. Ramanan. CATER: A diagnostic dataset for Compositional Actions TEmporal Reasoning. In *International Conference on Learning Representations*, 2020.

G. Glavaš and I. Vulić. Non-Linear Instance-Based Cross-Lingual Mapping for Non-Isomorphic Embedding Spaces. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7548–7555. Association for Computational Linguistics, 2020.

D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained Transformers Improve Out-of-Distribution Robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751. Association for Computational Linguistics, 2020.

X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174. Association for Computational Linguistics, 2020.

D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8018–8025, 2020.

C. Joshi. Transformers are Graph Neural Networks. *The Gradient*, 2020.

References

A. Kalinowski and Y. An. A Comparative Study on Structural and Semantic Properties of Sentence Embeddings. *CoRR*, abs/2009.11226, 2020.

L. Kong, C. de Masson d'Autume, L. Yu, W. Ling, Z. Dai, and D. Yogatama. A Mutual Information Maximization Perspective of Language Representation Learning. In *International Conference on Learning Representations*, 2020.

B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li. On the Sentence Embeddings from Pre-trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130. Association for Computational Linguistics, 2020a.

C. Li, X. Gao, Y. Li, X. Li, B. Peng, Y. Zhang, and J. Gao. Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space. 2020b.

G. Marcus. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. *CoRR*, abs/2002.06177, 2020.

R. T. McCoy, R. Frank, and T. Linzen. Does Syntax Need to Grow on Trees? Sources of Hierarchical Inductive Bias in Sequence-to-Sequence Networks. *Transactions of the Association for Computational Linguistics*, 8:125–140, 2020a.

R. T. McCoy, J. Min, and T. Linzen. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227. Association for Computational Linguistics, 2020b.

J. Min, R. T. McCoy, D. Das, E. Pitler, and T. Linzen. Syntactic Data Augmentation Increases Robustness to Inference Heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352. Association for Computational Linguistics, 2020.

B. Pan, Y. Yang, K. Liang, B. Kailkhura, Z. Jin, X. Hua, D. Cai, and B. Li. Adversarial Mutual Information for Text Generation. In *ICML*, 2020.

T. Schick and H. Schütze. Rare Words: A Major Problem for Contextualized Embeddings and How to Fix it by Attentive Mimicking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pages 8766–8774. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6403.

O. Sharir, B. Peleg, and Y. Shoham. The Cost of Training NLP Models: A Concise Overview. *CoRR*, abs/2004.08900, 2020.

A. Słowik, A. Gupta, W. L. Hamilton, M. Jamnik, S. B. Holden, and C. Pal. Exploring Structural Inductive Biases in Emergent Communication. *CoRR*, abs/2002.01335, 2020.

M. Treviso and A. F. T. Martins. The Explanation Game: Towards Prediction Explainability through Sparse Communication. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 107–118. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.blackboxnlp-1.10.

B. Wang, S. Wang, Y. Cheng, Z. Gan, R. Jia, B. Li, and J. Liu. InfoBERT: Improving Robustness of Language Models from An Information Theoretic Perspective. *CoRR*, abs/2010.02329, 2020. URL https://arxiv.org/abs/2010.02329.

A. Warstadt, Y. Zhang, X. Li, H. Liu, and S. R. Bowman. Learning Which Features Matter: RoBERTa Acquires a Preference for Linguistic Generalizations (Eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235. Association for Computational Linguistics, 2020.

I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454. Association for Computational Linguistics, 2020.

K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. CLEVRER: Collision Events for Video Representation and Reasoning. In *International Conference on Learning Representations*, 2020.

B. Zhang, I. Titov, and R. Sennrich. On Sparsifying Encoder Outputs in Sequence-to-Sequence Models. *CoRR*, abs/2004.11854, 2020a.

Y. Zhang, A. Warstadt, H.-S. Li, and S. R. Bowman. When Do You Need Billions of Words of Pretraining Data? *CoRR*, abs/2011.04946, 2020b.

L. Bauer, L. Deng, and M. Bansal. ERNIE-NLI: Analyzing the Impact of Domain-Specific External Knowledge on Enhanced Representations for NLI. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 58–69. Association for Computational Linguistics, 2021.

E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, page 610–623. Association for Computing Machinery, 2021.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

A. Goyal and Y. Bengio. Inductive Biases for Deep Learning of Higher-Level Cognition. *CoRR*, abs/2011.15091, 2021.

T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. 2021.

A. Lazaridou, A. Kuncoro, E. Gribovskaya, D. Agrawal, A. Liska, T. Terzi, M. Gimenez, C. de Masson d'Autume, S. Ruder, D. Yogatama, K. Cao, T. Kocisky, S. Young, and P. Blunsom. Pitfalls of Static Language Modelling. *CoRR*, abs/2102.01951, 2021.

T. Li, X. Chen, S. Zhang, Z. Dong, and K. Keutzer. Cross-Domain Sentiment Classification With Contrastive Learning and Mutual Information Maximization. In *ICASSP*, 2021.

References

C. Lovering, R. Jha, T. Linzen, and E. Pavlick. Predicting Inductive Biases of Pre-Trained Models. In *International Conference on Learning Representations*, 2021.

R. K. Mahabadi, Y. Belinkov, and J. Henderson. Variational Information Bottleneck for Effective Low-Resource Fine-Tuning. In *International Conference on Learning Representations*, 2021.

A. F. T. Martins. Reconciling the Discrete-Continuous Divide: Towards a Mathematical Theory of Sparse Communication. *CoRR*, abs/2104.00755, 2021.

C. Meister, S. Lazov, I. Augenstein, and R. Cotterell. Is Sparse Attention more Interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 122–129. Association for Computational Linguistics, 2021.

J. Su, J. Cao, W. Liu, and Y. Ou. Whitening Sentence Representations for Better Semantics and Faster Retrieval. *CoRR*, abs/2103.15316, 2021.

J. Wei, C. Meister, and R. Cotterell. A Cognitive Regularizer for Language Modeling. *CoRR*, abs/2105.07144, 2021.

W. Yu, C. Zhu, Y. Fang, D. Yu, S. Wang, Y. Xu, M. Zeng, and M. Jiang. Dict-BERT: Enhancing Language Model Pre-training with Dictionary. *ArXiv*, 2021.

L. Zhang, V. Prokhorov, and E. Shareghi. Unsupervised Representation Disentanglement of Text: An Evaluation on Synthetic Datasets. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 128–140. Association for Computational Linguistics, 2021.

T. Zhang and T. Hashimoto. On the Inductive Bias of Masked Language Modeling: From Statistical to Syntactic Dependencies. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5131–5146. Association for Computational Linguistics, 2021.

A

Introducing Relational Inductive Bias to Word Embeddings with Knowledge Graph

A.1 Intrinsic Evaluation of Knowledge Graph Embeddings

To verify the reliability of node2vec vector representations, we carried out an experiment on three standard word similarity datasets: RG-65 (Rubenstein and Goodenough, 1965), WordSim-353 similarity subset (Agirre et al., 2009), and SimLex-999 (Hill et al., 2015b). Table A.1 reports Pearson and Spearman correlations for the KG embedding technique (on WordNet's graph) and, as the baseline, for our two word embeddings, i.e. W2V-GN and GLOVE. We note that the performances are close to those of state-of-the-art WordNet approaches (Pilehvar and Navigli, 2015), which shows the efficacy of these embedding techniques in capturing the semantic properties of WordNet's graph.

KG/Word	**RG-65**		**WSS-353**		**SimLex-999**	
Embedding	ρ	r	ρ	r	ρ	r
node2vec	0.82	0.83	0.65	0.67	0.36	0.39
W2V-GN	0.75	0.77	0.77	0.76	0.44	0.45
GLOVE	0.76	0.75	0.66	0.66	0.37	0.39

Table A.1 Pearson (r) and Spearman (ρ) correlation results on three word similarity datasets.

B

Introducing Relational Inductive Bias to Sentence Embeddigns with Knowledge Graphs

B.1 DAGs

Graphs	**Mult.P$_{\%}$**	**AV.P**
PATO	31.29	2.97
$WN_{animal.n.01}$	0.88	2.00
$WN_{plant.n.02}$	0.16	2.00
HDO	16.23	2.13
HPO	23.24	2.23
GO	64.01	2.77
$WN_{entity.n.01}$	1.91	2.03

Table B.1 Statistics of nodes with multiple inheritances. Mult.P$_{\%}$ stands for the percentage of nodes with more than one parent node. AV.P stands for the average number of parents a node with multiple inheritance has.

B.1.1 Invalid Sequences

Graphs	Invalid$_{\%}$	N$_{\text{total}}$
PATO	1.82	110
WN$_{\text{animal.n.01}}$	4.56	263
WN$_{\text{plant.n.02}}$	2.23	314
HDO	4.02	622
HPO	7.08	847
GO	6.94	1845
WN$_{\text{entity.n.01}}$	8.50	5191

Table B.2 Statistics of invalid sequences. Invalid$_{\%}$ is the percentage of invalid sequences and N$_{\text{total}}$ is the total number of sequences that were tested.

B.2 Settings for Models

BOW-LR. To represent a KG in a vector space we use node2vec https://snap.stanford.edu/node2vec/. For all the graphs the following hyper-parameters of the algorithm are the same: *walk-length= 5*, *window-size=5* and *iter=40*. As for the number of dimensions we set it to 128 for PATO, WN$_{\text{animal.n.01}}$, WN$_{\text{plant.n.02}}$, HDO and HPO graphs. For GO and WN$_{\text{entity.n.01}}$ graphs we set it to 256. All the other parameters of node2vec are default.

We do not modify the numberbatch embeddings https://github.com/commonsense/conceptnet-numberbatch. If a word in a textual definition is missing we initilised the embedding for this word with zeros.

For all the graphs to map the textual vector space into a KG vector space we use the linear regression model from the *scikit-learn* API https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

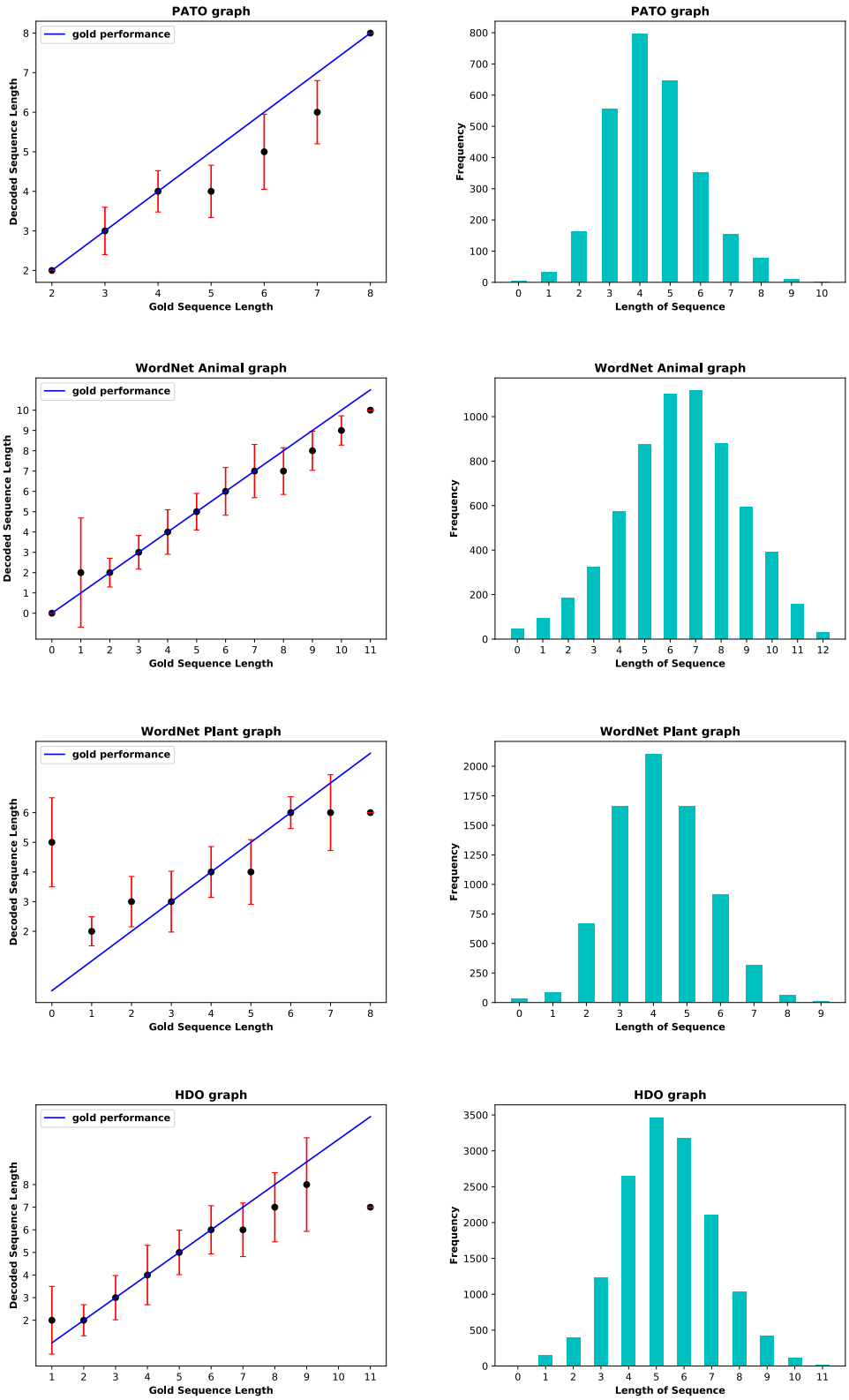


Fig. B.1 On the left graphs show: length of gold sequence vs mean length of decoded sequence on a test set; On the right graphs show: length of sequence vs length frequency on a training set.

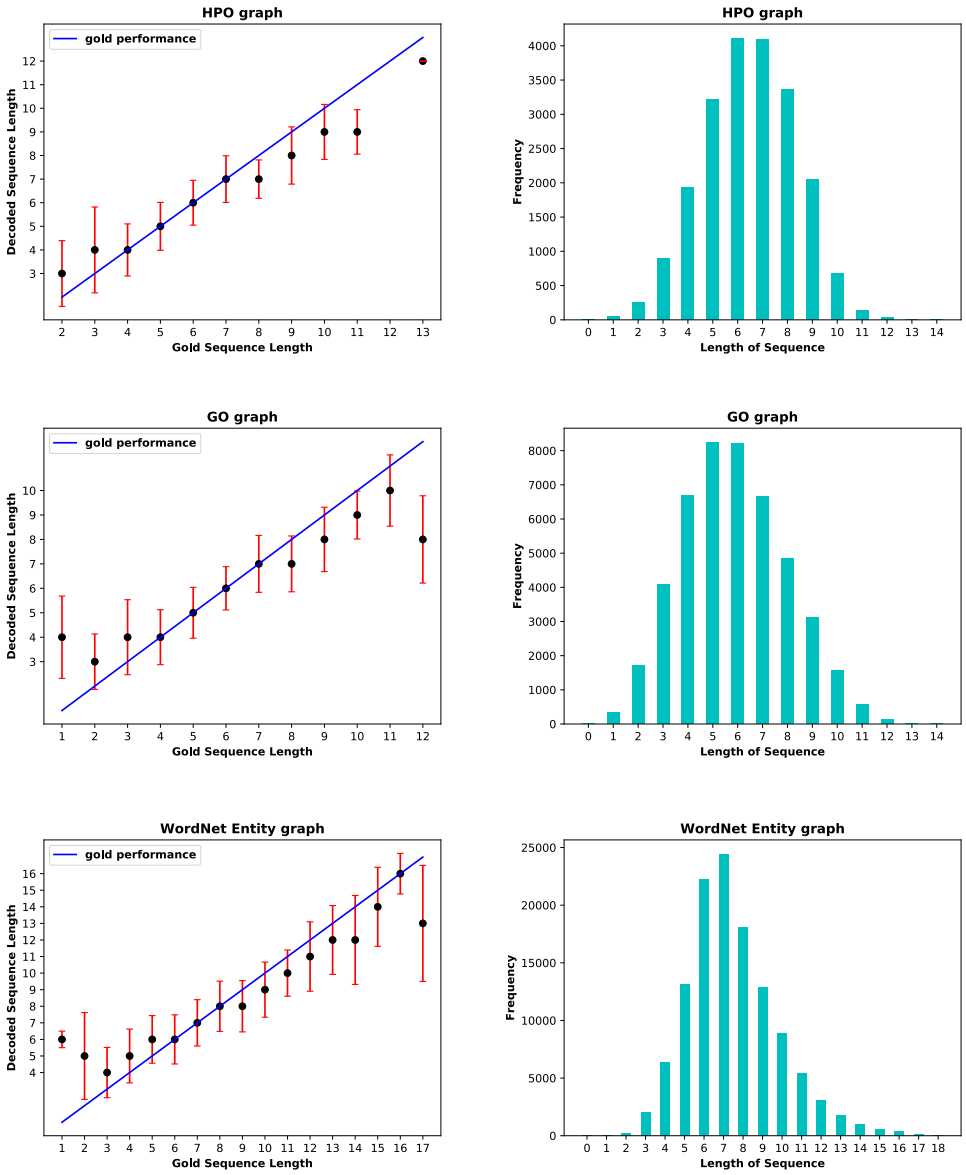


Fig. B.2 Continuation of Figure B.1. On the left graphs show: length of gold sequence vs mean length of decoded sequence on a test set; On the right graphs show: length of sequence vs length frequency on a training set.

MS-LSTM. There are only two hyper-parameters that we vary during the embedding of KG concepts: λ (we report the values in the chapter) and the embedding size of the concepts. We set it to 128 for PATO, $\text{WN}_{\text{animal.n.01}}$, $\text{WN}_{\text{plant.n.02}}$, HDO and HPO graphs. For GO and $\text{WN}_{\text{entity.n.01}}$ graphs we set it to 256.

For all the graphs the model is trained for 300 epochs, dimension of word embeddings is set to 64 and bi-LSTM is used instead of LSTM. Batch size is set to 16 and the number of latent dimensions in bi-LSTM is set to 128 for the PATO, $WN_{animal.n.01}$, $WN_{plant.n.02}$, HDO and HPO graphs. For GO and $WN_{entity.n.01}$ graphs we set these parameters to 128 and 256 respectively. All the other hyper-parameters are default.

When we use pre-trained word embeddings we reduce (with PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) its dimensions from 300 to 64.

Our Model. For all the graphs the model is trained for 300 epochs, dimensions of word embeddings (also for node/edges embeddings) is set to 64 and bi-LSTM is used in the encoder and LSTM in the decoder. Batch size is set to 16 and the number of latent dimensions in bi-LSTM encoder and LSTM decoder is set to 128 for the PATO, $WN_{animal.n.01}$, $WN_{plant.n.02}$, HDO and HPO graphs. For GO and $WN_{entity.n.01}$ graphs we set these parameters to 128 and 256 respectively. For optimizer we used *RMSProp* (https://www.tensorflow.org/api_docs/python/tf/train/RMSPropOptimizer) with learning rate = 0.001.

When we use pre-trained word embeddings we reduce (with PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html) its dimensions from 300 to 64.

B.3 Length of Generated Path

In Figures B.1 and B.2 the blue line indicates the ideal scenario i.e. mean length of the generated sequences is equal to the gold length. The black dot is the mean of the length of decoded sequences and the red bars are the standard deviation. One can notice that the general trend is the following: for short sequences the model generates (slightly) longer sequences and for the long sequences it generated (slightly) shorter sequences than the gold standard. Another trend is that the sequences of the certain length are matching the gold standard. To understand why this is happening one needs to look at the graph which relates the length of the sequence in the training corpus and the frequency of this length in the corpus. It is clear that there is a correlation between the two. Such as the model tends to generate the sequence of the length that is presented the most in the training data.

C

Learning Sentence Embeddings with VAE (Information-Theoretic Inductive Bias)

C.1 Statistics of Corpora

	Yelp	**DBpedia**	**Yahoo**
# sent. (train corpus)	100K	140K	100K
# sent. (valid corpus)	10K	14K	10K
# sent. (test corpus)	10K	14K	10K
vocabulary size	19,997	20K	20K
min sent. length.	20	1	5
av. sent. length.	96	35	12
max. sent. length.	200	60	30
# classes	5	14	10
# sent. in each class (train/test corpus)	20K/2K	10K/1K	10K/1K

Table C.1 Statistics of corpora. Vocabulary size excludes the ⟨pad ⟩and ⟨EOS ⟩symbols.

D

Learning Sentence Embeddings with VAE (Sparsity Inductive Bias)

D.1 Derivations of ELBO

Starting from the $\mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x))$, we derive the Evidence Lower Bound (ELBO) as follows:

$$\mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \int_{z,\gamma} dz d\gamma q_\phi(z,\gamma|x) \log \frac{q_\phi(z,\gamma|x)}{p_\theta(z,\gamma|x)}, \tag{D.1}$$

after rearranging terms in equation D.1 we can obtain:

$$\log p_\theta(x) - \mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \underbrace{\int_{z,\gamma} dz d\gamma q_\phi(z,\gamma|x) \log \frac{p_\theta(z,\gamma,x)}{q_\phi(z,\gamma|x)}}_{\text{ELBO}}, \tag{D.2}$$

Based on the independence assumption that we make in our graphical model (Figure 1) the generative model factorises as: $p_\theta(z,\gamma,x) = p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)$ and the inference model factorises as: $q_\phi(z,\gamma|x) = q_\phi(z|\gamma,x)q_\phi(\gamma|x)$. Therefore, we can rewrite the ELBO as follows:

$$\int_{z,\gamma} dz d\gamma q_\phi(z|\gamma,x)q_\phi(\gamma|x) \log \frac{p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)}{q_\phi(z|\gamma,x)q_\phi(\gamma|x)}, \tag{D.3}$$

We can further rewrite the ELBO as a sum of the three separate terms. Where the first term is:

$$\begin{aligned}
&\int_{z,\gamma} dz d\gamma q_\phi(z|x,\gamma) q_\phi(\gamma|x) \log p_\theta(x|z) \\
&\int_{\gamma} d\gamma q_\phi(\gamma|x) \int_{z} dz q_\phi(z|x,\gamma) \log p_\theta(x|z) \therefore \\
&\left\langle \int_{z} dz q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} \therefore
\end{aligned} \tag{D.4}$$

The second term is:

$$\begin{aligned}
&\int_{z,\gamma} dz d\gamma q_\phi(z|x,\gamma) q_\phi(\gamma|x) [\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)] \\
&\left\langle \int_{z} dz q_\phi(z|x,\gamma) [\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)] \right\rangle_{q_\phi(\gamma|x)} \therefore \\
&\left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} \therefore
\end{aligned} \tag{D.5}$$

Finally, the third term is:

$$\begin{aligned}
&\int_{z,\gamma} dz d\gamma q_\phi(z|x,\gamma) q_\phi(\gamma|x) [\log q_\phi(\gamma|x) - \log p_\theta(\gamma)] \\
&\int_{\gamma} d\gamma q_\phi(\gamma|x) [\log q_\theta(\gamma|x) - \log p_\theta(\gamma)] \underbrace{\int_{z} dz q_\phi(z|x,\gamma)}_{\text{sums to 1 for each:}\gamma} \therefore \\
&\int_{\gamma} d\gamma q_\phi(\gamma|x) [\log q_\phi(\gamma|x) - \log p_\theta(\gamma)] \therefore \\
&\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)) \therefore
\end{aligned} \tag{D.6}$$

Collecting all the three terms into the single ELBO:

$$\left\langle \int_{z} dz q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} - \left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} - \mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)), \tag{D.7}$$

D.2 Objective Functions of Mathieu et al. (2019) and Tonolini et al. (2019)

The objective function of Mathieu et al. (2019) is:

$$\left\langle \log p_\theta(x|z) \right\rangle_{q_\phi(z|x)} - \psi KL(q_\phi(z|x)||p_\theta(z)) - \lambda \mathbb{D}(q_\phi(z), p_\theta(z)),$$

where ψ and λ are the scalar weight on the terms and Tonolini et al. (2019) is:

$$\left\langle \log p_\theta(x|z) \right\rangle_{q_\phi(z|x)} - KL(q_\phi(z|x)||q_\phi(z|x_u) - J \times \mathbb{D}_{KL}\left(\bar{\gamma}_u||\alpha\right)),$$

where J is the dimensionality of the latent variable z, x_u is a learnable pseudo-input (Tomczak and Welling, 2018) and α is prior sparsity.

D.3 Deriving Marginal of (Univariate) Spike-and-Slab Prior

We derive the Spike-and-Slab distribution by integrating out the index component which is distributed as a Bernoulli variable. This result is quite well-known in machine learning, however for the ease of the reader we present it here as a quick reference.

The derivation: assume 1) $\pi \sim p(\pi;\gamma)$ is a $Bernoulli(\gamma)$ and 2) $p(z|\pi) = (1-\pi) \times p_1(z) + \pi \times p_2(z)$, where $p_1(z) \sim N(z;0,1)$ and $p_2(z) \sim N(z;0,\sigma \to 0)$ is a Spike-and-Slab model. The the marginal Spike-and-Slab prior over z can be obtained in the following way:

$$p(z;\gamma) = \sum_{i=0}^{1} p(z|\pi = i)p(\pi = i;\gamma)$$

$$p(z|\pi = 0)p(\pi = 0;\gamma) + p(z|\pi = 1)p(\pi = 1;\gamma) \therefore$$

$$[(1-0) \times p_1(z) + 0 \times p_2(z)]p(\pi = 0;\gamma) + [(1-1) \times p_1(z) + 1 \times p_2(z)]p(\pi = 1;\gamma) \therefore$$

Expanding brackets:

$$p_1(z)p(\pi=0;\gamma)+p_2(z)p(\pi=1;\gamma)\therefore$$
$$N(z;0,1)p(\pi=0;\gamma)+N(z;0,\sigma\to 0)p(\pi=1;\gamma)\therefore$$
$$(1-\gamma)N(z;0,1)+\gamma N(z;0,\sigma\to 0)\therefore$$

Therefore,

$$p(z;\gamma)=(1-\gamma)N(z;0,1)+\gamma N(z;0,\sigma\to 0).$$

D.4 End-to-end Differentiable

Sampling a value from the Spike-and-Slab posterior distribution $q(z|x,\gamma)$ is a two step process. First a spike or slab component is sampled which is a binary decision, we use Binary Concrete distribution (Maddison et al., 2016) to make this sampling step end-to-end differentiable. Then the value is sampled from the corresponding component, for this we employ the reparameterisation trick (Kingma and Welling, 2014). Also, samples from the Beta distribution are pathwise differentiable (Figurnov et al., 2018).

D.5 Hoyer

This section reports Average Hoyer, for the two corpora Yelp and Yahoo, both on the mean and samples from the posterior distributions of the HSVAE and MAT-VAE models.

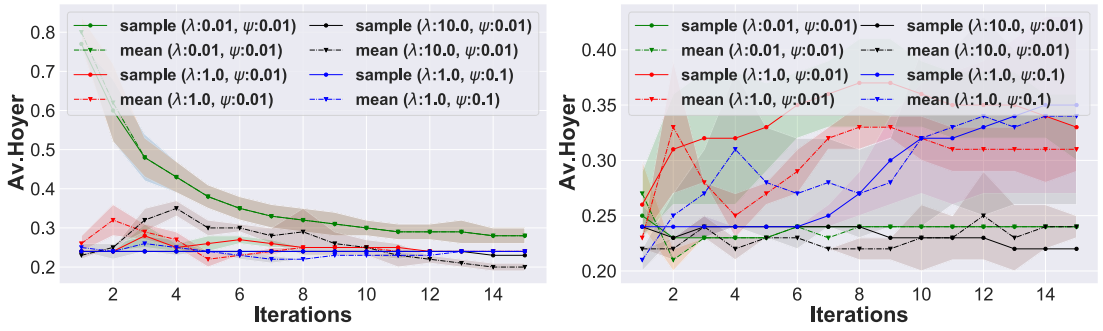


Fig. D.1 Average Hoyer (Av.Hoyer) on Yelp (left) and Yahoo (right) corpora dev set for MAT-VAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

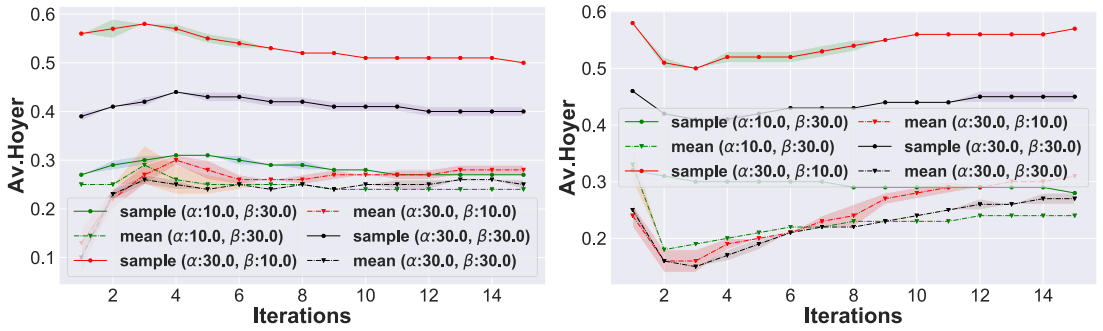


Fig. D.2 Average Hoyer (Av.Hoyer) on Yelp (left) and Yahoo (right) corpora dev set for HSVAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.